TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACP	Analyse en composantes principales
ANOVA	*Analysis of variance*
AUC	*Area under curve*
BML	*Boosted mixture learning*
BPSO	*Binary particle swarm optimisation*
CNN	*Convolutional neural network*
DET	*Detection error tradeoff*
EER	*Equal error rate*
EM	Espérance-maximisation
FRNN	*Fuzzy relational neural network*
FSVM	*Fuzzy support vector machine*
GMM	*Gaussian mixture-model*
GRNN	*General regression neural network*
HOSVD	*Higher-order singular value decomposition*
ILSVRC	*ImageNet large scale visual recognition challenge*
LLR	*Log-likelihood ratio*
LPC	*Linear prediction coefficients*
LPCC	*Linear prediction cepstral coefficients*
LSTM	*Long-short term memory*
MFCC	*Mel frequency cepstral coefficients*
MLP	*Multilayer perceptron*
OLS	*Orthogonal least square*
PLP	*Perceptual linear prediction*

PNN	*Probabilistic neural network*
SVM	*Support vector machine*
TDNN	*Time-delay neural network*
UBM-GMM	*Universal background model gaussian mixture model*
WLPCC	*Weighted linear prediction cepstral coefficients*

LISTE DES SYMBOLES ET UNITÉS DE MESURE

s	seconde
ms	milliseconde
Hz	hertz
kHz	kilohertz

INTRODUCTION

La *Déclaration du Millénaire*, adoptée par l'Organisme des Nations Unies en septembre 2000, énonçait les *Objectifs du Millénaire*, une liste de huit objectifs à remplir avant 2015 (United Nations, 2000). Le quatrième de ces objectifs portait sur la mortalité infantile. En effet, l'une des cibles était de réduire de deux tiers, entre 1990 et 2015, le taux de mortalité des enfants de moins de 5 ans.

Bien que l'objectif n'ait pas été atteint en 2015, des progrès importants ont été réalisés, la quantité annuelle de décès d'enfants agés de moins de 5 ans étant passée de 9,8 millions en 2000 à 5,9 millions en 2015 (Unicef & World Health Organization, 2015). Néanmoins, une très grande proportion (45%) des victimes étaient des enfants âgés de moins d'un mois. Cette proportion a augmenté, elle était de 33% en 2000.

Suite à cette constatation, l'Organisation des Nations Unies a décidé d'accorder une attention plus particulière au taux de mortalité néonatale. Elle a adopté, en septembre 2015, les *Objectifs de développement durable*, une résolution de 17 nouveaux objectifs à remplir d'ici 2030, faisant suite aux *objectifs du Millénaire*. L'une des nouvelles cibles est de réduire le taux de mortalité néonatale à 12 pour 1000 naissances vivantes au plus, ainsi que le taux de mortalité des enfants de moins de 5 ans à 25 pour 1000 naissances vivantes au plus (United Nations, 2015). La santé néonatale est donc devenue l'une des préoccupations principales pour l'Organisation des Nations Unies.

Afin de guider les efforts dans la lutte à la mortalité infantile et de mesurer les progrès effectués, l'Organisation mondiale de la Santé produit annuellement un rapport qui recense les décès d'enfants, ainsi que leurs causes. Les estimations pour l'année 2016 sont résumées dans la figure 0.1.

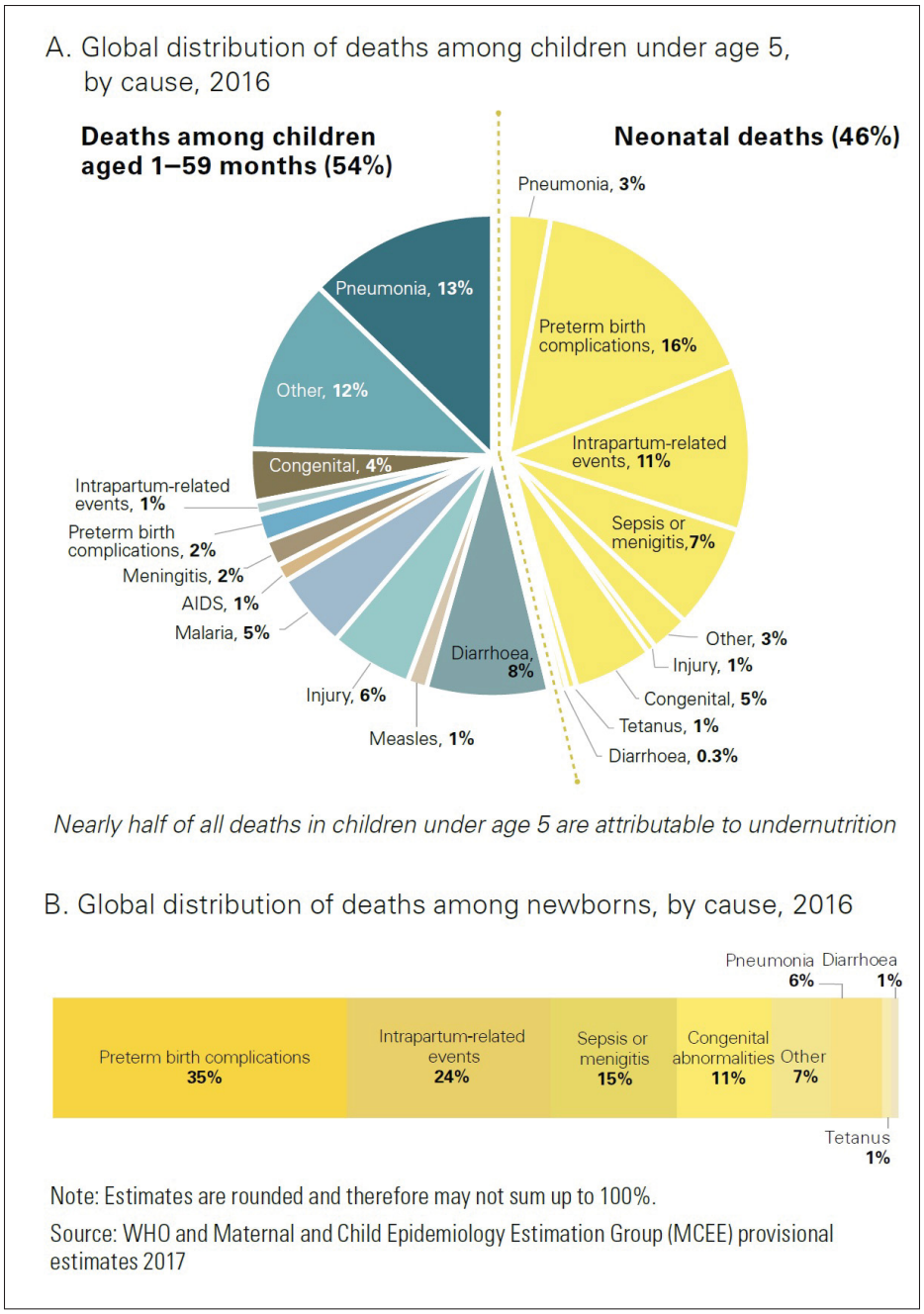

Figure 0.1 Causes globales de mortalité chez A. les enfants de moins de 5 ans et B. les nouveaux-nés, en 2016.
Figure tirée de Unicef *et al.* (2017, p.11)

Selon cette étude :

A. Les complications découlant de naissances prématurées sont la cause de mortalité néonatale principale, provoquant 35% des décès de nourrissons dans le monde. Les enfants nés prématurément viennent au monde avant d'avoir atteint le stade de développement normal. Ils sont donc plus susceptibles de souffrir, entre autres, de problèmes respiratoires (Fraser *et al.*, 2004), d'hypothermie (Jost *et al.*, 2017) et d'infections (McGuire *et al.*, 2004). Les survivants risquent de souffrir de troubles d'apprentissage et de problèmes visuels ou auditifs (World Health Organization, 2017) ;

B. Presque un quart (24%) des décès néonataux sont causés par des complications lors de la naissance. Il existe plusieurs complications possibles. L'une des plus communes est l'asphyxie périnatale. Une carence d'oxygène prolongée peut causer des dommages irréparables aux organes, en particulier le cerveau, et même la mort. Il est difficile de prévenir l'asphyxie périnatale. Les enfants qui y survivent auront cependant besoin de soins urgents, car leurs cerveaux, poumons, coeurs, reins, foies, entre autres, risquent d'être endommagés (Antonucci *et al.*, 2014) ;

C. Les maladies infectieuses sont la troisième cause de décès néonataux la plus importante. La sepsie, la méningite, la pneumonie et le tétanos sont ensembles responsables de près de 22% des décès de nourrissons. Ces pathologies doivent être traitées par antibiotiques (Shane *et al.*, 2017), (Kim, 2010), (Thwaites *et al.*, 2015), (Nissen, 2007) ;

D. Les anomalies congénitales sont une autre cause importante de mortalité (11% des décès). Les anomalies congénitales les plus communes sont les cardiopathies congénitales, les anomalies du tube neural et le syndrome de Down.

Le combat contre ces pathologies passe tout d'abord par l'application de mesures préventives. Assurer une gestation saine est l'un des moyens les plus importants. Les mères souffrant de carences alimentaires (World Health Organization, 2010), faisant usage de drogues (World

Health Organization, 2010) (Goldenberg *et al.*, 2008), atteintes de maladies transmises sexuellement (Nissen, 2007), (World Health Organization, 2010) ou d'infections vaginales (Goldenberg *et al.*, 2008) ont plus de chances de mettre au monde des enfants prématurés ou atteints de pathologies néonatales. Vacciner la mère, s'assurer que l'accouchement a lieu dans un milieu propre et aseptisé, forcer le lavage des mains, nettoyer le conduit vaginal avant chaque examen etc. sont toutes des mesures qui permettent aussi de réduire les risques (World Health Organization, 2010), (Nissen, 2007), (Thwaites *et al.*, 2015), (Shane *et al.*, 2017), (McGuire *et al.*, 2004).

Lorsque la prévention n'a pas suffi, et qu'un enfant souffre d'une pathologie dès la naissance, des soins sont requis. La rapidité de l'intervention est cruciale à la survie aussi bien qu'à la qualité de vie de l'enfant atteint. En général, l'administration rapide du traitement réduit les risques de complications et augmente les chances de survie. De même, un traitement rapide permettrait de réduire la gravité des séquelles ou même de les éviter complètement, épargnant à l'enfant de souffrir de handicaps. Dans le même ordre d'idées, les enfants affligés de handicaps physiques ou mentaux qui ne reçoivent pas l'accompagnement approprié auront plus de difficulté à s'adapter et prendront du retard dans leur développement (Bos *et al.*, 2013), (Volkmar, 2014), (Johnson, 2017). Il est important de les prendre en charge le plus tôt possible, pour les aider à surmonter leurs limitations et à être fonctionnels en société. C'est un avantage pour les enfants, leur famille, ainsi que la société en général.

Pour pouvoir intervenir, la pathologie doit d'abord être détectée, puis identifiée. Le dépistage et le diagnostic précoce sont donc d'une importance capitale .

Les techniques de dépistage et de diagnostic existantes sont souvent intrusives, coûteuses, et requièrent l'expertise de professionnels qualifiés. L'analyse automatisée de cris de nourrisson a le potentiel de fournir un outil non-intrusif et simple à utiliser, qui pourrait certainement être utile dans la lutte à la mortalité et la morbidité néonatale.

En effet, au cours du 20ième siècle, des pédiatres se rendirent compte que plusieurs pathologies avaient un effet important sur les caractéristiques acoustiques et spectrales des cris, ce qui les porta à considérer le potentiel de l'analyse des cris pour le diagnostic. La lecture des spectrogrammes est cependant une tâche compliquée et quelque peu subjective, que très peu de médecins maîtrisent. Les techniques de diagnostic spectrographiques sont donc sous-utilisées en médecine (Golub & Corwin, 1985).

L'automatisation de la reconnaissance des caractéristiques spectrographiques, ainsi que de leur interprétation, se présente donc comme une perspective intéressante, qui pourrait aider à répandre ce nouvel outil de diagnostic.

Le domaine de l'ingénierie biomédicale s'est donc intéressé à l'analyse et la classification automatisée des cris de nourrissons. Les pathologies ayant les effets les plus évidents sur le cri, comme la surdité et l'asphyxie périnatale, ont fait l'objet de beaucoup de tentatives.

Beaucoup d'études cherchent à créer un outil de diagnostic capable de reconnaître une ou deux pathologies. Étant donné que plusieurs pathologies partagent des symptômes vocaux similaires (variations de la durée et de la fréquence fondamentale des cris, etc.), il nous semble plus intéressant de commencer par poursuivre le développement d'un outil de dépistage, qui ne pose pas de diagnostic précis mais permet de repérer rapidement les nourrissons qui requièrent un examen plus approfondi. La tâche consiste simplement à reconnaître si l'enfant est atteint d'au moins une pathologie dans la liste des pathologies ciblées.

En général, les systèmes d'analyse automatisée sont basés sur l'apprentissage machine supervisé. Ils sont construits de la façon suivante :

A. Plusieurs cris provenant d'enfants dont l'état de santé est connu sont enregistrés ;

B. Un algorithme de traitement de signal est utilisé pour extraire les caractéristiques des cris pertinentes à la reconnaissance des pathologies ;

C. Un classifieur est entraîné à distinguer les caractéristiques des cris pathologiques de ceux d'enfants en santé. Certains exemples de cris servent à entraîner le classifieur, tandis que d'autres servent à mesurer sa performance.

Il existe une multitude d'algorithmes de classification. Plusieurs types de classifieurs ont déjà été évalués pour cette tâche (comme les modèles de mélange gaussien (GMM), les machines à vecteurs de support (SVM) et divers types de réseaux de neurones). Au cours de cette étude, nous nous sommes intéressés aux réseaux de neurones artificiels, plus particulièrement à deux nouvelles architectures : les réseaux convolutionnels et les réseaux récurrents.

Les réseaux convolutionnels sont des réseaux de neurones spécialisés dans le traitement d'images. Ils ont obtenu d'énormes succès dans le traitement et la reconnaissance d'images (mentionnons Krizhevsky *et al.* (2012) et Szegedy *et al.* (2015), entre autres). Étant donné qu'un spectrogramme peut être considéré comme une image, nous avons décidé de tester cette nouvelle architecture sur notre sujet.

Les réseaux récurrents, quant à eux, sont des réseaux de neurones spécialisés dans le traitement de séquences. Ils ont permis de grandes avancées dans plusieurs domaines, comme la modélisation du langage et la traduction (citons Sutskever *et al.* (2014), Wu *et al.* (2016) par exemple). Étant donné qu'un spectrogramme peut être considéré comme une séquence de spectres, nous avons choisi de tester cette architecture aussi.

Notons cependant que ces deux architectures sont nées de l'apprentissage profond, un nouveau paradigme de développement de réseaux de neurones, qui est caractérisé par un nombre très élevé de couches (d'où son nom). L'entraînement des énormes réseaux ainsi produits est rendu possible par l'explosion récente de la quantité de données et de ressources de calcul disponibles. Dans le cadre de ce projet, nous n'avons accès qu'à peu de données et de ressources

de calcul. Nous nous limitons donc à des réseaux d'une seule couche cachée, même si nous utilisons des architectures issues de l'apprentissage profond.

Comme on a pu le constater, le dépistage automatisé par analyse de cris de nourrissons combine plusieurs domaines, notamment l'acoustique, la pédiatrie, le traitement de signal et l'apprentissage machine.

Dans le premier chapitre de ce mémoire, nous abordons le fonctionnement du système vocal chez l'humain, en particulier chez le nourrisson. Nous résumons également plusieurs recherches pédiatriques portant sur les caractéristiques des cris de nourrissons atteints de diverses pathologies. Pour finir, nous révisons plusieurs études sur la reconnaissance automatisée de pathologies par analyse de cris de nourrissons.

Dans le deuxième chapitre, nous révisons brièvement les concepts de traitement de signal et d'apprentissage machine utiles au développement de notre système de diagnostic automatisé. Les composantes utilisées dans la construction de nos réseaux de neurones sont présentés, ainsi que les techniques utilisées pour les entraîner.

Dans le troisième chapitre, nous décrivons les architectures testées, les expériences réalisées et les outils utilisés au cours de ce projet.

Dans le quatrième et dernier chapitre, les résultats obtenus sont présentés et discutés.

CHAPITRE 1

CRIS DE NOURRISSONS

1.1 Modèle physio-acoustique

1.1.1 Système vocal humain

La production de la parole est un procédé biomécanique complexe, qui requiert le contrôle précis de plusieurs muscles.

Le système vocal se subdivise en 3 sous-systèmes : Les systèmes subglottal, glottal et supra-glottal. Nous décrirons brièvement le rôle de chacun de ces systèmes ainsi que leur fonctionnement.

1.1.1.1 Système subglottal

Le système subglottal est constitué des poumons et de la trachée. Il contribue à la phonation en poussant un flux d'air à travers le larynx. Chez les adultes, plusieurs mesures indiquent que la pression alvéolaire est maintenue stable lors de la phonation. Cette régulation de la pression nécéssite un contrôle complexe des muscles intercostaux. Des mesures indirectes indiquent qu'il en est probablement de même chez le nourrisson, même si le mécanisme de régulation est légèrement différent (Lieberman, 1985). Le rythme des inspirations et expirations, qui est normalement régi par un système de régulation végétatif, doit également être contrôlé. Chez l'adulte, la quantité d'air inspirée est ajustée en fonction de la durée prévue de la phonation. Chez le nourrisson, le contrôle du rythme respiratoire est plus simple. Plusieurs mesures indiquent que les expirations ont tendance à durer environs 5 fois plus longtemps que les inspirations (Lieberman, 1985). Notez que la pression de l'air poussé par les poumons affecte aussi indirectement la fréquence fondamentale (Zhang, 2016).

1.1.1.2 Système glottal

La glotte correspond à l'ouverture entre les cordes vocales, qui sont situées dans le larynx. Les sons voisés sont produits lorsque les cordes vocales entrent en vibration et modulent le flux d'air qui les traversent. La fréquence des impulsions ainsi produites est nommée la fréquence fondamentale, ou $F0$. La fréquence fondamentale est contrôlée en ajustant la tension, la rigidité et la longueur des cordes vocales, ainsi que l'ouverture de la glotte, à l'aide de différents muscles (Zhang, 2016).

1.1.1.3 Système supraglottal

Le système supraglottal sert de caisse de résonnance. Il est constitué du pharynx et de la cavité orale. De plus, lorsque le velum est ouvert, la cavité nasale est couplée au tractus vocal (Rabiner & Juang, 1993). La réponse fréquentielle du système supraglottal est réglée en ajustant précisément la position de plusieurs articulateurs (la langue, la machoire, le velum, les lèvres, etc.). On appelle parfois "formants" les fréquences de résonnance du système supraglottal.

Pour ajouter à la complexité de la production du cri chez les nouveaux-nés, il semble qu'une rétroaction auditive soit également impliquée, selon les études de Cullen *et al.* (1967), Eilers & Oller (1994), Clement & Beinum (1995) et Möller & Schönweiler (1999).

Le cri chez le nouveau-né est donc un phénomène complexe, régulé par rétroaction auditive, requiérant un contrôle précis du système respiratoire, du larynx et de plusieurs articulateurs, comme la langue, la mâchoire, etc. Il suffit qu'une pathologie vienne débalancer l'un de ces mécanismes pour que le son produit soit affecté. Par exemple, des maladies respiratoires peuvent influencer certains paramètres du cri, comme la durée, la puissance ou le rythme. (Golub & Corwin, 1985). De même, des atteintes au système nerveux peuvent affecter le contrôle glottal et l'articulation, modifiant ainsi la fréquence fondamentale ou les formants (Golub & Corwin, 1985), (Wasz-Höckert *et al.*, 1985). Des problèmes auditifs peuvent également avoir des effets sur les cris (Eilers & Oller, 1994), (Clement & Beinum, 1995), (Möller & Schönweiler, 1999).

On peut donc espérer que l'analyse de cris permette de déceler la présence d'anomalies chez les nouveau-nés, et peut-être même d'en identifier précisémment la cause (Golub & Corwin, 1985).

1.1.2 Modélisation

Le modèle source-filtre permet de modéliser la phonation de façon simplifiée. Un exemple de phonation produite par le modèle source-filtre est présenté à la figure 1.1.

Dans cet exemple, tiré de (Golub & Corwin, 1985), le flux d'air produit par le système glottal est la somme d'une source périodique $S(t)$ et d'une source de bruit $N(t)$.

Ce signal est filtré par le système supraglottal, dont la réponse fréquentielle est $T(f)$, puis par une fonction de transfert représentant le filtrage entre la bouche et le micro, dont la réponse fréquentielle est $R(f)$. Le spectre du signal capté par le micro est donc :

$$O(f) = R(f)T(f)\left[S(f) + N(f)\right] \tag{1.1}$$

La fonction de transfert représentant le filtrage entre la bouche et le micro est appelée "caractéristique de radiation".

Notez que le modèle source-filtre ne permet pas de modéliser la production de tous les sons pouvant être émis par le système vocal humain.

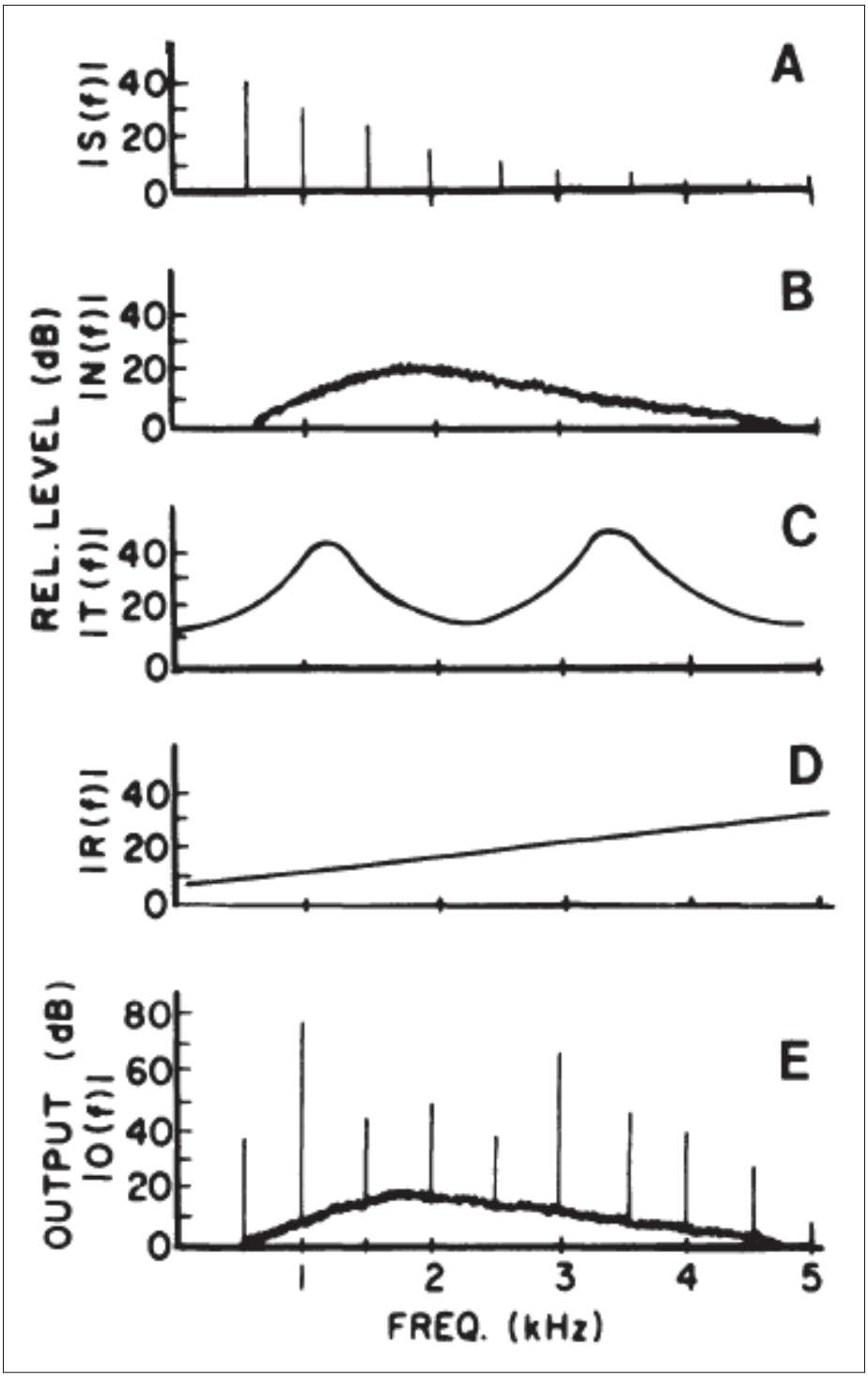

Figure 1.1 A. Spectre d'une source périodique idéale ; B. Spectre d'une source de bruit idéale ; C. Réponse fréquentielle d'un système supraglottal idéal ; D. Réponse fréquentielle d'une caractéristique de radiation idéale ; E. Spectre du son produit, selon le modèle.
Exemple tiré de Golub & Corwin (1985, p.64)

1.2 Analyse spectrographique

Au cours du 20ième siècle, l'analyse spectrographique fut rendue possible grâce à l'invention du magnétophone, qui permettait d'enregistrer les cris, et du et, qui permettait de calculer les spectrogrammes des enregistrements. L'apparation de l'ordinateur vint ensuite faciliter davantage l'utilisation de cette technique.

Grâce à ces nouveaux outils, les cris de nourrissons purent être étudiés en profondeur. Plusieurs études révèlèrent que certains paramètres acoustiques des cris de nourrissons étaient affectés par la présence de pathologies. De plus, il devint clair que l'occurence anormale de certains motifs spectrographiques dans les cris était fortement corrélée avec la présence de certaines pathologies précises.

Wasz-Hockert et son équipe furent parmi les premiers à définir les paramètres et motifs pertinents pour la reconnaissance de pathologies. Les plus importantes de ces caractéristiques sont présentées dans les sections suivantes.

1.2.1 Paramètres acoustiques pertinents

1.2.1.1 Durée

Wasz-Höckert *et al.* (1985) définirent la durée d'un cri comme le temps écoulé entre le début du cri et la fin de la dernière phonation avant l'inspiration suivante. Plusieurs études indiquent que la durée du cri est affectée par certaines pathologies. Ainsi, la durée moyenne des cris d'enfants atteints du syndrome de Down, par exemple, est de 4.5 secondes (Lind *et al.*, 1970), contre 2.6 pour les cris de douleurs d'enfants en santé (Wasz-Höckert *et al.*, 1968).

1.2.1.2 Fréquence fondamentale (F0)

Wasz-Höckert *et al.* (1985) caractérisent la plage de variation de la fréquence fondamentale en mesurant la moyenne des $F0$ minimum et maximum sur plusieurs cris. Les $F0$ maximums

moyens mesurés dans les études de Wasz-Höckert *et al.* (1968), Ostwald *et al.* (1968), Michelsson (1971) et Thodén & Koivisto (1980) variaient entre 540 Hz et 680 Hz. Les $F0$ minimums moyens variaient entre 330 Hz et 420 Hz. Plusieurs pathologies ont un effet marqué sur la fréquence fondamentale. Par exemple, Wasz-Höckert *et al.* (1971) ont mesuré un $F0$ maximum moyen de 2120 Hz et un $F0$ minimum moyen de 960 Hz dans les cris d'enfants souffrant d'hyperbilirubinémie.

1.2.1.3 Latence

Lorsque le cri est élicité par un stimuli douloureux, la latence correspond au temps écoulé entre le stimuli et le début du cri. Une étude par Fisichelli & Karelitz (1963) indique que la latence est plus élevée qu'à la normale chez les enfants ayant souffert de dommages au cerveau.

1.2.1.4 Fréquences de résonnance

Les études de Colton & Steinschneider (1981), Corwin *et al.* (1995) et Robb *et al.* (2013) révèlèrent toutes que les cris d'enfants décédés du syndrome de la mort subite présentaient des fréquences de résonnances anormalement basses ou élevées.

1.2.2 Motifs spectrographiques

1.2.2.1 Mélodie

La mélodie décrit l'évolution temporelle de la fréquence fondamentale. Wasz-Höckert *et al.* (1985) distinguent 5 types de mélodies :

- Mélodie montante (**M**),
- Mélodie tombante (**T**),
- Mélodie montante-tombante (**MT**),
- Mélodie tombante-montante (**TM**),

- Mélodie plate (**P**).

Les mélodies tombantes et montante-tombantes sont les plus communes dans le cris d'enfants en santé. Les mélodies plates sont fréquentes dans les cris d'enfants souffrant d'anomalies chromosomiques (Vuorenkoski *et al.*, 1966), Lind *et al.* (1970), (Michelsson *et al.*, 1980). Les mélodies montantes et tombante-montante sont associées aux cris d'enfants atteints au système nerveux central (Michelsson *et al.*, 1977a), (Michelsson *et al.*, 1977b).

1.2.2.2 *Double harmonic break*, *biphonation*, *furcation*

Un *double harmonic break* réfère à l'apparition de lignes parralèles sur le spectrogramme, entre la fréquence fondamentale et ses harmoniques. Ce motif est commun dans les cris de douleur (Wasz-Höckert *et al.*, 1968).

La *biphonation* correspond à l'apparition de deux fréquences fondamentales sur le spectrogramme, accompagnées de leurs harmoniques respectives. Les deux séries d'harmoniques n'évoluent pas de façon parrallèle. La *biphonation* est très rare dans les cris d'enfants en santé (Wasz-Höckert *et al.*, 1968), mais peut apparaître fréquemment dans les cris d'enfants atteints de certaines pathologies qui atteignent le système nerveux central (Michelsson, 1971), (Michelsson *et al.*, 1977a), (Michelsson *et al.*, 1977b), (Michelsson *et al.*, 1982).

La *furcation* décrit la séparation de la fréquence fondamentale en plusieurs fondamentales. On l'observe surtout des les cris d'enfants atteints d'hyperbilirubinémie (Wasz-Höckert *et al.*, 1971).

1.2.2.3 *Glottal Roll* et *vibrato*

Un *Glottal Roll* est un son ayant une fréquence fondamentale très basse. Il est fréquent à la fin des cris. Il est parfois précédé par un *vibrato*. Le *vibrato* est une ondulation de la fréquence fondamentale. Ce sont des motifs communs dans les cris de nourrissons en santé (Wasz-Höckert *et al.*, 1968), (Michelsson, 1971), (Thodén & Koivisto, 1980).

1.2.2.4 *Noise concentration*

Un *Noise concentration* réfère à la présence, dans les parties voisées comme non-voisées du cri, d'un pic d'énergie autour de 2000Hz-2500Hz. Le *Noise concentration* a été observé dans les cris d'enfants souffrant d'une encéphalite au virus herpès simplex (Pettay *et al.*, 1977).

1.2.3 Symptômes spectrographiques de certaines pathologies

Maintes études ont été menées sur les effets que certaines pathologies ont sur ces paramètres et motifs. Le tableau 1.1, inspiré d'une liste dressée par Wasz-Höckert *et al.* (1985), présente les symptômes vocaux de plusieurs pathologies.

Dans ce tableau, une flèche vers le haut peut soit indiquer l'augmentation de la valeur d'un paramètre ou l'augmentation de la probabilité d'occurence d'un motif. Une flèche vers le bas signifie l'inverse.

Ainsi, certaines pathologies sont caractérisées par la présence d'un motif précis. L'encéphalite au virus herpès simplex, par exemple, est trahie par la présence de *noise concentration*.

Une grande portion des pathologies partagent cependant des symptômes spectrographiques similaires. De plus, il est complexe de distinguer automatiquement certains motifs importants, comme le *double harmonic break*, la *biphonation* et la *furcation*.

Reconnaître précisément une pathologie en ne se basant que sur l'analyse des cris est donc difficile. Cette observation nous porte à croire que l'analyse des cris devrait tout d'abord servir comme outil de dépistage.

Tableau 1.1 Symptômes vocaux principaux de plusieurs pathologies

Groupe	Pathologie	F0	Autres symptômes principaux	Source
Anomalies chromosomiques	Chromosomes 4 & 5	↑	-	(Michelsson *et al.*, 1980)
	Cri du chat (chromosome 5)	↑	*Mélodie* ***P*** ↑, *Mélodie* ***M*** ↑	(Vuorenkoski *et al.*, 1966), (Luchsinger *et al.*, 1967), (Michelsson *et al.*, 1980)
	Chromosomes 13 & 18	↓	*Mélodie* ***P*** ↑	(Michelsson *et al.*, 1980)
	Syndrome de Down (chromosome 21)	↓	Durée ↑, *Mélodie* ***P*** ↑	(Lind *et al.*, 1970)
Perturbations du système endocrinien	Hypothyroïdie congénitale	↓	*Glottal roll* ↑, Cri rauque	(Michelsson & Sirviö, 1976)
Perturbations du métabolisme	Hyperbilirubinémie	↑	*Biphonation* ↑, *Furcation* ↑	(Wasz-Höckert *et al.*, 1971)
	Hypoglycémie	↑	*Biphonation* ↑, *Vibrato* ↑, *Glide* ↑	(Koivisto *et al.*, 1974), (Michelsson *et al.*, 1982)
Asphyxie	Asphyxie	↑	*Biphonation* ↑, *Glide* ↑ *Mélodie* ***M*** ↑ *Mélodie* ***TM*** ↑	(Michelsson, 1971)
Pathologies du système nerveux central	Méningite bactérienne	↑	Durée ↓ *Biphonation* ↑, *Glide* ↑ *Mélodie* ***M*** ↑ *Mélodie* ***TM*** ↑	(Michelsson *et al.*, 1977b)
	Encéphalite au virus herpès simplex	↑	*Biphonation* ↑, *Glide* ↑ *Noise concentration*	(Pettay *et al.*, 1977)
	Hydrocéphalie	↑	*Biphonation* ↑, *Glide* ↑ *Mélodie* ***P*** ↑	(Michelsson *et al.*, 1984)
Malnutrition	Marasme	↑	*Mélodie* ***P*** ↑	(Juntunen *et al.*, 1978)
Malformations	Maladie de Krabbe	↑	*Mélodie* ***M*** ↑ *Mélodie* ***TM*** ↑	(Thodén & Michelsson, 1979)

1.3 Classification automatisée de cris de nourrissons

1.3.1 Architecture générale

La classification automatisée des cris est généralement réalisée par apprentissage supervisé. Le système est construit de la façon suivante :

A. Une base de données d'enregistrements de cris est recueillie et annotée. À chaque échantillon est associé une étiquette, indiquant à quelle classe il appartient ;

B. Des algorithmes de traitement de signal sont utilisés pour extraire les attributs de chaque échantillon audio. Les attributs sont des caractéristiques pertinentes à la reconnaissance de la classe d'un échantillon ;

C. Un algorithme est parfois utilisé pour réduire le nombre d'attributs. Dans certaines situations, cela permet de faciliter l'apprentissage et d'obtenir de meilleures performances ;

D. Un classifieur est entraîné à reconnaître la classe des échantillons à partir de leurs attributs. L'entraînement est réalisé à partir d'échantillons provenant de la base de données, ainsi que leurs étiquettes.

1.3.2 Premières tentatives d'automatisation

Petroni *et al.* (1995) furent, à notre connaissance, les premiers à tenter de développer un système automatisé de classification de cris de nourrissons. Ils entraînèrent différents types de réseaux à reconnaître 3 émotions : La peur, la faim et la douleur. Ils ne considérèrent que la première seconde de la première uttérance suivant le stimulis. Ils éliminèrent les uttérances de moins de 0.75 secondes. Ils découpèrent la première seconde de l'uttérance en 125 trames de 16ms, avec un recouvrement de 50%, puis calculèrent les coefficients du cepstre de fréquences mel (MFCC) de chaque trame. Les uttérances de moins d'une seconde étaient rallongées jusqu'à une seconde en répliquant la dernière trame. Ils obtinrent une précision de classification maximale de 77.9%, démontrant que la reconnaissance automatisée d'émotions dans les cris

de nourrissons était possible. Leur échantillons provenaient d'enregistrements de cris collectés à l'hôpital de Montréal pour enfants. Ils utilisèrent la validation croisée à 10 *folds* pour valider les modèles.

1.3.3 Reconnaissance de pathologies

1.3.3.1 Recherche mexicaine

Dans les années 2000, un groupe de chercheurs mexicains, menés par le professeur Carlos Alberto Reyes García, réalisa plusieurs études sur le diagnostic automatisé de pathologies par analyse de cri de nouveau-nés.

Ils commencèrent par récolter des enregistrements de cris d'enfants malentendants et d'enfants ayant souffert d'asphyxie. Ils recueillèrent également des cris d'enfants en santé, séparés en trois catégories : Cris de douleur, cris de faim et cris "normaux".

Les enregistrements rassemblés forment la base de données aujourd'hui connue sous le nom de *Baby Chillanto Database*.

Une fois la majorité des échantillons recueillis, ils commencèrent à utiliser cette base de données pour entraîner des classifieurs.

Orozco-García & Reyes-García (2003b) s'inspirèrent des travaux de Petroni *et al.* (1995) pour mettre au point une procédure d'extraction d'attributs qui fut ensuite réutilisée dans pratiquement toutes le recherches de ce groupe, avec des variations mineures. Cette procédure est décrite ci-dessous :

A. Les enregistrements de cris d'enfants sont segmentés en échantillons d'une durée variant de 0.4 à 3 secondes ;

B. Chaque échantillon est ensuite séparé en trames d'une durée 50 ou 100 millisecondes, puis une transformée est appliquée sur chacune des trames, afin de produire les attributs de l'échantillon. Deux types de transformée sont considérés. L'une d'entre elles produit

les coefficients du cepstre de fréquences mel (MFCC). L'autre produit les coefficients de prédiction linéaire (LPC) ;

C. L'analyse en composantes principales (ACP) est souvent utilisée par la suite pour réduire le nombre d'attributs.

Les MFCC surpassèrent les LPC dans les études de Orozco-García & Reyes-García (2003a), Orozco-García & Reyes-García (2003b) et Reyes-Galaviz & Reyes-García (2004). De plus, Orozco-García & Reyes-García (2003b) obtinrent des performances égales ou supérieures lorsqu'ils firent passer le nombre de MFCC de 21 à 16. Les études subséquentes utilisèrent donc les 16 MFCC comme attributs.

Le groupe de recherche se basa sur ces attributs standards pour concentrer leurs efforts sur le classifieur. Ils utilisèrent la base de données *Baby Chillanto* pour entraîner plusieurs types de modèles à remplir diverses tâches similaires.

Orozco-García & Reyes-García (2003a) et Orozco-García & Reyes-García (2003b) utilisèrent des percerptrons multicouches (MLP) pour reconnaître les enfants atteints de surdité. Ils utilisèrent la validation croisée à 10 *folds* et obtinrent une précision maximale de 97.43%.

Reyes-Galaviz & Reyes-García (2004) et Reyes-Galaviz *et al.* (2005) entraînèrent quant à eux des TDNN (*time delay neural network*) à reconnaître trois classes : les enfants malentendants, les enfants ayant souffert d'asphyxie et les enfants en santé. Ils obtinrent une précision maximale de 98.67%, en validant leur modèle par la méthode *holdout* (Goodfellow *et al.*, 2016).

Reyes-Galaviz *et al.* (2008) effectuèrent la même expérience, mais pour une tâche légèrement différente : il n'y avait que deux classes, l'une contenant les cris d'enfants souffrants de surdité ou d'asphyxie et l'autre des cris d'enfants en santé. Ils obtinrent une précision de 100%. Par contre, lorsqu'ils entraînèrent leur système sur une autre base de données, provenant de Cuba, ils obtinrent des résultats nettement inférieurs. Aussi, ils utilisèrent un algorithme évolutif pour réduire la taille des attributs et obtinrent de bien meilleurs résultats qu'avec l'ACP.

Suaste-Rivas *et al.* (2004a) et Suaste-Rivas *et al.* (2004b) se tournèrent vers la logique floue. Ils entraînèrent des FRNN (*fuzzy relational neural networks*) à reconnaître trois classes : les enfants malentendants, les enfants ayant souffert d'asphyxie et les enfants en santé. L'espace de caractéristiques était divisé en 7 termes linguistiques. Parmi les fonctions d'appartenance testées, la fonction trapézoïdale produisit les meilleurs résultats, avec une précision de 98%. Les modèles furent validés par la méthode *holdout*.

Rosales-Pérez *et al.* (2011) utilisèrent également des FRNN, mais se servirent d'un algorithme génétique pour optimiser les hyperparamètres du réseau, y compris les fonctions d'appartenance et le nombre de termes linguistiques. Ils entraînèrent les FRNN à remplir trois tâches de classification binaire différentes : reconnaissance de l'asphyxie, reconnaissance de la surdité et distinction entre les cris de faim et les cris de douleur. Ils validèrent leurs modèles par validation croisée à 10 *folds* et obtinrent des précisions maximales de 88.67%, 97% et 96.03% respectivement.

Barajas-Montiel & Reyes-García (2006) entraînèrent des SVM (*support vector machines*) et des FSVM (*fuzzy support vector machine*) et à reconnaître trois classes : les enfants malentendants, les enfants ayant souffert d'asphyxie et les enfants en santé. Les FSVM obtinrent une précision de 94.9816% contre 94.7741% pour les SVM normales. Les modèles furent validés par validation croisée à 10 *folds*.

Santiago-Sánchez *et al.* (2009) entraînèrent un algorithme d'appariement de formes basé sur les *type-2 fuzzy sets* (T2-FPM) sur deux tâches de classification : La reconnaissance de l'hyperbilirubinémie et de l'asphyxie (classication trinaire) et la reconnaissance de l'hyperbilirubinémie seule (classification binaire). Pour ce qui est des attributs, ils utilisèrent la procédure standard, avec les MFCC et les LPC, mais évaluèrent également le cochléogramme et l'intensité calculés par le logiciel Praat. Ils n'utilisèrent pas l'ACP. Pour la première tâche, la meilleure précision obtenue était de 91.74%, en utilisant la combinaison des LPC et du cochléogramme. Pour la seconde tâche, la meilleure précision obtenue était de 95.56%, en utilisant la combinaison des MFCC et des LPC. Leur modèles étaient validés par validation croisée à 10 *folds*.

Comme on a pu le constater, ce groupe de recherche s'intéressait principalement au classifieur. Ils définirent donc une procédure d'extraction d'attributs standard, qui fut réutilisée par la suite sans beaucoup de variations, et comparèrent différents modèles entraînés avec la base de données *Baby Chillanto* à accomplir différentes tâches similaires. Tous les types de classifieurs utilisés, c'est-à-dire des MLP, TDNN, FRNN, SVM, FSVM et T2-FPM, obtinrent de bonnes performances.

En plus des classifieurs, ce groupe étudia un peu la sélection des attributs. La majorité des études se servirent de l'ACP pour réduire le nombre d'attributs. Une approche évolutionnaire de sélection des attributs fut également testée dans Reyes-Galaviz *et al.* (2008).

1.3.3.2 Recherche malaisienne

Un groupe de chercheurs malaisiens s'est aussi intéressé au développement de système de diagnostic automatisé par analyse de cris de nourrissons. De leur côté, ils ont surtout concentré leurs efforts sur le design et la sélection des attributs. Ils se sont principalement servi de deux bases de données : La base de données mexicaine *Baby Chillanto*, ainsi que la base de données de l'université *Milano-Bicocca*, en Italie.

Les premières études de ce groupe utilisèrent les attributs standards développés par le groupe de recherche mexicain et se concentrèrent sur les algorithmes de sélection des attributs. En effet, un nombre d'attributs trop élevé peut parfois compliquer la tâche du classifieur et nuire à l'apprentissage. Il vaut mieux dans ce cas sélectionner un sous-ensemble des attributs les plus pertinents à la tâche.

Sahak *et al.* (2010), Sahak *et al.* (2012), Zabidi *et al.* (2010) utilisèrent l'algorithme OLS (*orthogonal least squares*) pour trier les MFCC en fonction de leur importance. Leurs résultats montrèrent tous que la performance pouvait être augmentée avec la sélection OLS. Ils travaillèrent tous sur la reconnaissance de l'asphyxie, mais les types de modèles et les bases de données utilisées variaient.

Zabidi *et al.* (2011) comparèrent deux algoithmes de sélection d'attributs : L'un était basé sur le *F-ratio*, l'autre était de type BPSO (*binary particle swarm optimisation*). Ils entraînèrent des MLP à reconnaître l'asphyxie. L'algorithme de type BPSO permit d'obtenir de meilleures performances avec moins de neurones dans la couche cachée. Les exemples d'enfants en santé provenaient de la base de données *Baby Chillanto* et ceux d'enfants ayant souffert d'asphyxie provenaient de la base de données de l'université *Milano-Bicocca*.

Wahid *et al.* (2016) comparèrent 6 autres algorithmes de sélection d'attributs, pour trois tâches de classification différentes : la reconnaissance de l'asphyxie, la reconnaissance de la surdité et la distinction entre les cris de douleur et les cris de faim. Ils varièrent également les types d'attributs et de modèles utilisés. L'algorithme *OneR* permit d'obtenir les meilleures performances dans les deux tâches de reconnaissance de pathologies, alors que l'algorithme *ReliefF* obtint les meilleures performances dans la tâche de reconnaissance de type de cri. Les données utilisées provenaient de la base de données *Baby Chillanto*.

Les études de ce groupe de recherche portèrent également sur la conception d'autres types d'attributs.

Dans leur étude, Wahid *et al.* (2016) comparèrent les attributs de type MFCC et LPCC (*Linear prediction cepstral coefficients*), avec et sans leurs dérivées première et secondes. Les quatres combinaisons d'attributs donnèrent des performances semblables. Les modèles étaient validés par validation croisée à 10 *folds*.

Hariharan *et al.* (2012a) de leur côté comparèrent les attributs LPC (*linear prediction coefficients*), LPCC (*linear prediction cepstral coefficients* et WLPCC (*weighted linear prediction cepstral coefficients*). Ils utilisèrent la base de données *Baby Chilanto* pour entraîner des PNN (*probabilistic neural networks*) à reconnaître l'asphyxie et la surdité. Les modèles furent validés par la méthode *holdout*. Les WLPCC obtinrent les meilleurs résultats et atteignirent une précision de plus de 98%.

Hariharan *et al.* (2011) utilisèrent l'énergie et l'entropie de Shannon de la transformée en paquet d'ondelettes comme attributs. Les ondelettes de Daubechies furent utilisées, avec un nombre variable de moments dissipants. Le niveau de décomposition fut varié de 1 à 5. Ils utilisèrent la base de données *Baby Chillanto* pour entraîner des PNN à reconnaître l'asphyxie. Aux niveaux de décomposition plus faibles, les attributs d'énergie obtinrent de meilleures performances, tandis qu'aux niveaux de décomposition plus élévés, les attributs d'entropie obtinrent des performances légèrement supérieures. La décomposition sur 5 niveaux avec les ondelettes d'un seul moment dissipant suffit pour atteindre une précision de plus de 95%. La décomposition sur 5 niveaux avec les ondelettes de 20 moments dissipants permit d'atteindre des précisions de plus de 99%. Les PNN furent validés avec la méthode *holdout*.

Hariharan *et al.* (2012b) et Saraswathy *et al.* (2013) formèrent leurs attributs en regroupant plusieurs statistiques sur quatre graphiques dérivés du spectrogramme :

- Le graphique TF (*Time Frequency*), constitué du spectrogramme de densité de puissance ;
- La courbe TMA (*Time Maximum Amplitude*), décrivant, pour chaque trame, l'amplitude maximum dans les fréquences ;
- La courbe FMA (*Frequency Maximum Amplitude*), décrivant, pour chaque bande de fréquence, l'amplitude maximum dans le temps ;
- La courbe FSDA (*Frequency Standard Deviation Amplitude*), décrivant, pour chaque bande de fréquence, l'écart-type dans le temps.

Ils testèrent leurs attributs en entraînant divers types de modèles avec des échantillons provenant de la base de données *Baby Chillanto*. Hariharan *et al.* (2012b) entraînèrent des MLP, TDNN et GRNN (*general regression neural networks*) à reconnaître la surdité. Ils rapportent une précision de plus de 99% avec le modèle GRNN, évaluée par validation croisée à 10 *folds*. Saraswathy *et al.* (2013) de leur côté entraînèrent des PNN et des GRNN à reconnaître la surdité et l'asphyxie (classification trinaire). Ils commencèrent par effectuer une analyse de variance (ANOVA), qui révéla que chaque attribut avait une grande importance discriminatoire (avec

des *p-values* de moins de 0.001). Ils rapportent une précision de plus de 99% avec le modèle PNN, évaluée par validation croisée à 10 *folds*.

Ainsi, les chercheurs malaisiens travaillèrent principalement sur le design ainsi que la sélection des attributs. Les premières recherches utilisèrent les attributs standards développés par le groupe de recherche mexicain pour comparer plusieurs algorithmes de sélection des attributs. D'autres procédures d'extraction d'attributs furent développées et testées dans des recherches ultérieures. Certaines obtinrent d'excellentes performances.

1.3.3.3 Autres études

Farsaie Alaie & Tadj (2012) Farsaie Alaie & Tadj (2013), Farsaie Alaie *et al.* (2016) étudièrent la reconnaissance de pathologies chez les nouveaux-nés basée sur les modèles de mélange gaussien (GMM).

Les GMM sont constitués d'une somme pondérée de gaussiennes. Ils peuvent former une approximation lisse à n'importe quelle distribution (Reynolds, 2008).

Ils peuvent être utilisés pour construire un classifieur. Il suffit d'utiliser un GMM différent pour modéliser la distribution des attributs des échantillons de chaque classe. Ainsi, pour un échantillon donné, chaque GMM retourne la probabilité que l'échantillon appartienne à la classe qu'il modélise. Plusieurs techniques différentes peuvent ensuite être employées pour effectuer la décision finale à partir des probabilités estimées.

Farsaie Alaie & Tadj (2012) entraînèrent un classifieur GMM à reconnaître deux classes de trames : Les trames de cris d'enfants sains et celles provenant de cris d'enfants atteints de pathologies. La classe pathologique comportait plusieurs pathologies rares. Chaque classe contenait des cris d'enfants prématurés aussi bien que ceux d'enfants nés à terme. À des fins de comparaison, les paramètres des GMM de chaque classe furent estimés à partir des données d'entraînement par deux algorithmes différents : l'algorithme EM (espérance-maximisation) et l'algorithme BML (*boosted mixture learning*, un algorithme qui sélectionne automatiquement

le nombre de gaussiennes dans la mixture). La décision était effectuée par le critère du *Maximum Likelihood* (La classe choisie est celle dont le GMM retourne la plus haute probabilité). Les 13 MFCC des trames furent utilisés comme attributs. Le modèle fut validé par la méthode *holdout*. Farsaie Alaie & Tadj (2013) réalisèrent exactement la même expérience, mais entraînèrent leur modèle à reconnaître 8 classes : une pour les enfants sains nés à terme, une pour les enfants sains nés prématurément et six pour six pathologies rares. Dans ces deux études, l'algorithme BML obtint de meilleures performances que l'algorithme EM. Leurs échantillons provenaient de la base de données de l'École de Technologie Supérieure, à Montréal.

Farsaie Alaie *et al.* (2016) tentèrent de bâtir un système capable de reconnaître les cris d'enfants malades, puis de détecter le système affecté. Ils groupèrent les pathologies étudiées en 5 groupes, selon le système affecté : respiratoire, cardiaque, neurologique, sanguin ou autre. Ils subdivisèrent le problème en deux tâches. La première tâche était de distinguer les segments de cris provenant d'enfants malades de ceux provenant d'enfants en santé. La seconde tâche était de reconnaître le système affecté chez les enfants reconnus comme malades lors de la première tâche. Les 13 MFCC des trames, accompagnés de leurs dérivées première et seconde, servaient d'attributs. Ils utilisèrent la base de données de l'École de Technologie Supérieure.

La première tâche fut réalisée par fusion de deux classifieurs GMM, l'un modélisant les segments expiratoires, l'autre les segments inspiratoires. Les paramètres des GMM furent obtenus par adaptation d'un UBM-GMM (*universal background model gaussian mixture model*), qui modélisait tous les segments, peu importe qu'ils proviennent d'un enfant malade ou en santé. Les paramètres des UBM-GMM (l'un modélisant les segments expiratoires, l'autre les segments inspiratoires) furent estimés par l'algorithme EM. L'adaptation des paramètres fut effectuée par 4 méthodes différentes. Le logarithme du rapport de vraisemblance (LLR) fut ensuite calculé sur les probabilités retournées par chacun des modèles. Deux versions différentes du LLR furent évaluées. Un autre classifieur servit ensuite à effectuer la décision finale en se basant sur les LLR retournés par les modèles inspiratoires et expiratoires. La meilleure précision ainsi obtenue était de 91.68%.

Pour ce qui est de la seconde tâche, qui consistait à identifier le système atteint chez le nourrisson, seuls les pathologies neurologiques et respiratoires furent considérées, car la quantité de données pour les autres pathologies était insuffisante. Le modèle fut construit de façon similaire à celui de la première tâche. Les UBM-GMM ne modélisaient que les segments de cris provenant d'enfants atteints des pathologies étudiées. Cette seconde étape obtint de bien moins bonnes performances.

Une autre étude intéressante est celle de Chittora & Patil (2015). Ils calculèrent le bispectre des trames voisées de cris de nourrissons. Ils comparèrent ensuite 4 méthodes pour extraire l'information pertinente du bispectre et former les attributs. Ils évaluèrent leurs attributs en entraînant des SVM à reconnaître les cris d'enfants malades, atteints de l'une des pathologies suivantes : infection des voies respiratoires supérieures, septicémie, malnutrition, chirurgie, épilepsie, diarrhée, hydrocéphalie, hypocalcémie, cardiopathie congénitale, jaunisse et bronchite. Leurs échantillons provenaient de la base de données de l'hôpital civil d'Ahmedabad, en Inde. Parmi les 4 méthodes testées, la décomposition en valeurs singulières d'ordre supérieur (HOSVD) permit d'obtenir les meilleurs résultats. Ils entraînèrent également les SVM avec les attributs MFCC, LPC et PLP (*perceptual linear prediction*), à des fins de comparaison. Les attributs basés sur la décomposition en valeur singulières d'ordre supérieur du bispectre produisirent les meilleurs résultats, avec une précision de 70%. Les performances furent validées avec la méthode *holdout*.

1.3.3.4 Résumé

Ainsi, les groupes de recherche mexicains et malaisiens rapportèrent les meilleures performances. Cependant, comme il sera discuté à la section 4.4.1, il est possible qu'une bonne partie de ces études comporte des biais de validation, à cause d'une partition inadéquate des échantillons en ensembles d'entraînement et de validation. La plupart de ces études utilisaient exactement les mêmes attributs, de type MFCC. Quelques études malaisiennes développèrent d'autres types de caractéristiques, certaines basées sur les paquets d'ondelettes, d'autres sur les statistiques des spectrogrammes, d'autres sur les LPCC, etc.

L'étude de Farsaie Alaie *et al.* (2016), basée sur les GMM-UBM, produisit aussi de bon résultats, qui ne devraient pas souffrir du même biais de validation, car leurs données furent adéquatement partitionnées en ensembles d'entraînement et de validation. Ce détail est discuté davantage à la section 4.7.

Dans l'étude de Chittora & Patil (2015), les attributs basés sur le bispectre produisirent une faible précision de classification (70%), qui surpassait toutefois les attributs MFCC et LPC. Les attributs MFCC et LPC obtinrent des précisions largement inférieures (52% et 61% respectivement) à celles rapportées par les études mexicaines et malaisiennes.

Il est possible que la difficulté de la tâche réalisée par Chittora & Patil (2015) explique cette différence. En effet, beaucoup de pathologies, très différentes les unes des autres, sont représentées dans leur classe pathologique. Si c'était le cas, cela signifierait que les attributs basés sur le bispectre sont supérieurs.

La présence de biais de validation dans les études mexicaines et malaisiennes, mais pas dans celle de Chittora & Patil (2015) pourrait également causer une telle différence de performances. Il est cependant difficile de déterminer si la méthode de partition employée dans l'étude de Chittora & Patil (2015) était adéquate.

Un total de 5 bases de données furent utilisées dans ces études :

A. La base de données de l'hôpital de Montréal pour enfants,
B. La base de données *Baby Chillanto*,
C. La base de données *Milano-Bicocca*,
D. La base de données de l'École de Technologie Supérieure,
E. La base de données de l'hôpital civil d'Ahmedabad.

Toutes ces bases de données contiennent des enregistrements de cris d'enfants en santé et d'enfants atteints de diverses pathologies. Le nombre d'enfants et d'échantillons pour chaque pa-

thologie est en général très faible, ce qui ne facilite pas la modélisation et limite la signification statistique des performances mesurées.

Dans le cadre de ce projet, nous nous sommes inspirés des attributs standards MFCC développés par les chercheurs mexicains, ainsi que des attributs MFCC de Farsaie Alaie *et al.* (2016). Nous entraînons deux nouveaux types de réseaux de neurones, les réseaux convolutionnels (CNN) et les réseaux récurrents *Long short-term memory* (LSTM), à reconnaître diverses pathologies. Nous entraînons également des perceptrons multicouches (MLP) à des fins de comparaison. Nous nous servons de la base de données de l'École de Technologie Supérieure.

CHAPITRE 2

THÉORIE

Dans ce chapitre, nous présentons la théorie pertinente à la compréhension de ce projet. Nous expliquons d'abord ce que sont les coefficients du cepstre mel. Nous présentons ensuite plusieurs notions d'apprentissage machine, ainsi que divers types de réseaux de neurones et un algorithme pour les entraîner.

2.1 Coefficients du cepstre mel (MFCC)

2.1.1 Échelle mel et spectre mel

L'échelle mel, développée par Stevens *et al.* (1937), permet de mesurer la hauteur tonale perçue par l'oreille humaine. Plusieurs équations représentent le lien entre la fréquence f, en hertz, et la hauteur tonale m, en mels. L'une des plus populaires est celle de O'Shaughnessy (1987) :

$$m = 2595 \log_{10}(1 + f/700) \tag{2.1}$$

Le spectre mel, quant à lui, vise à représenter le spectre perçu par l'oreille humaine. Il est calculé en multipliant le spectre du signal par une banque de filtres triangulaires. Ces filtres sont séparés linéairement et partagent la même largeur de bande sur l'échelle mel. Sur l'échelle des fréquences, cela produit des filtres dont l'espacement et la largeur de bande augmentent exponentiellement avec la fréquence. Cette banque de filtres est fréquemment approximée en utilisant des filtres de largeur et d'espacement linéaire sous 1000 Hz. Au dessus de 1000 Hz, la fréquence centrale de chaque filtre est 1.1 fois supérieure à celle du filtre précédent, donnant des largeurs et espacement exponentiels. La figure 2.1 montre un exemple de banque de filtres mel.

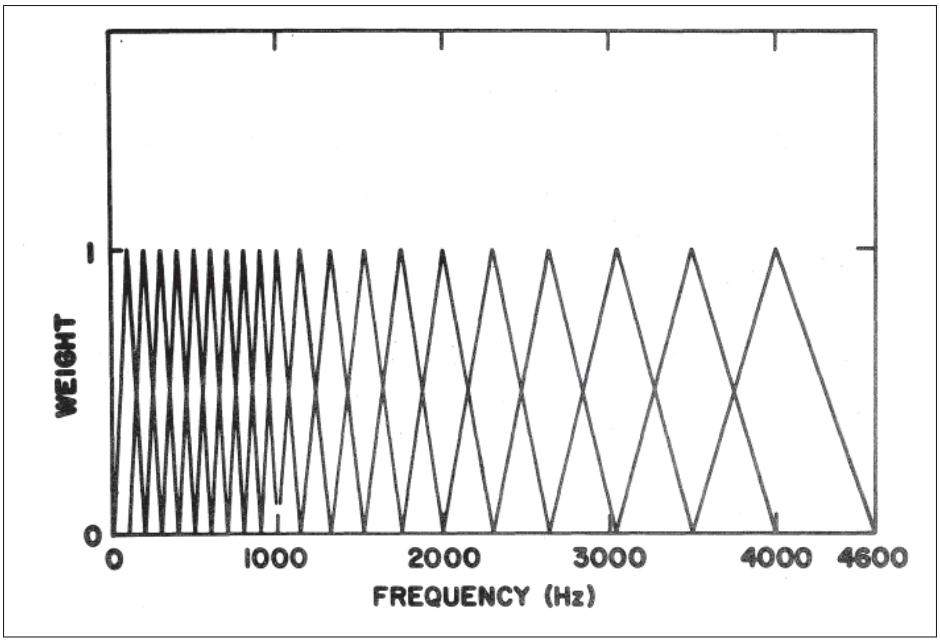

Figure 2.1 Exemple de banque de filtres mel,
tiré de Davis & Mermelstein (1980, p.200)

2.1.2 Cepstre

Le calcul du cepstre est une opération de déconvolution homomorphique (Oppenheim & Schafer, 2004). Un opérateur de déconvolution $D[\,]$ a comme propriété :

$$D\left[x_1(n) * x_2(n)\right] = D[x_1(n)] + D[x_2(n)] \tag{2.2}$$

Cette opération peut être réalisée en calculant le spectre du logarithme du spectre, qu'on appelle cepstre. En effet, si deux signaux sont convolués, leurs spectres sont multipliés.

$$\mathscr{F}\left[x_1(n) * x_2(n)\right] = \mathscr{F}\left[x_1(n)\right]\mathscr{F}\left[x_2(n)\right] = X_1(k)X_2(k) \tag{2.3}$$

Par la suite, l'application du logarithme permet de transformer le produit de spectres en somme de logarithme de spectres.

$$\log[X_1(k)X_2(k)] = \log[X_1(k)] + \log[X_2(k)] \tag{2.4}$$

L'application de la transformée de fourier inverse permet ensuite d'aller dans le domaine cepstral.

$$\mathscr{F}^{-1}[\log[X_1(k)] + \log[X_2(k)]] = \mathscr{F}^{-1}[\log[X_1(k)]] + \mathscr{F}^{-1}[\log[X_2(k)]] \tag{2.5}$$

Cette opération est utilisée dans plusieurs domaines pour étudier les signaux qui ont été produits par convolution. Elle fut proposée la première fois par Bogert *et al.* (1963) pour la détection des échos. Le cepstre fut ensuite utilisé dans plusieurs autres applications, comme par exemple l'extraction de la fréquence fondamentale de signaux de parole (Noll, 1967).

Le cepstrogramme, quant à lui, est constitué des cepstres de plusieurs trames consécutives. Il permet d'étudier l'évolution du cepstre dans le temps.

2.1.3 Cepstre de fréquences mel

Le cepstre de fréquence mel est calculé à partir du spectre mel, plutôt que du spectre standard. De plus, la transformée de fourier inverse est normalement remplacée par une transformée en cosinus discrète.

La transformée en cosinus discrète du signal vecteur $\boldsymbol{x}$ de longueur N est :

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right) \quad k = 0,1,\ldots,N-1 \tag{2.6}$$

Cette transformée permet de compresser le signal.

Les étapes pour calculer le cepstre de fréquence mel sont donc :

A. Calculer le spectre du signal,

B. Multiplier par une banque de filtres mel,

C. Calculer le logarithme,

D. Effectuer la transformée en cosinus discrète.

Les coefficients du cepstrogramme mel ainsi calculé sont couramment appelés MFCC (*Mel Frequency Cepstral Coefficients*).

2.1.4 Post-traitement

Un post-traitement est souvent appliqué sur les MFCC. Le *liftering* (Young *et al.*, 2013), par exemple, est parfois utilisé pour accentuer certains coefficients et en atténuer d'autres. Il est réalisé en multipliant le cepstre par une fenêtre. Une fenêtre sinusoïdale, paramétrisée par L, est souvent utilisée (Juang *et al.*, 1987). Chacun des poids w_i d'une telle fenêtre vaut :

$$w_i = \begin{cases} 1 + \frac{L}{2}\sin\left(\frac{i\pi}{L}\right) & i = 1, 2, \ldots L \\ 0 & \textit{sinon} \end{cases} \tag{2.7}$$

2.2 Apprentissage machine et réseaux de neurones

Cette section porte sur l'apprentissage machine, plus particulièrement sur les réseaux de neurones. La majorité des algorithmes à base de réseaux de neurones sont basés sur quatre éléments : un modèle, des données, une fonction de coût et une procédure d'entraînement (Goodfellow *et al.*, 2016). Ces quatres éléments sont présentés aux sous-sections 2.2.2, 2.2.3, 2.2.4 et 2.2.5, après que quelques notions générales aient été expliquées à la sous-section 2.2.1.

Les sous-sections 2.2.6 et 2.2.7 présentent ensuite les concepts importants de généralisation et de la validation, puis la sous-section 2.2.8 discute de l'optimisation des hyperparamètres.

2.2.1 Notions générales

2.2.1.1 Algorithmes d'apprentissage

Pour Mitchell (1997), on dit qu'un programme apprend de l'expérience E par rapport à une certaine classe de tâches T et mesure de performance P si sa performance aux tâches dans T, mesurée par P s'améliore avec l'expérience E.

Dans ce projet, nous effectuons une tâche de classification. Notre mesure de performance est la précision de la classification. L'expérience est constituée des échantillons de notre base de données.

2.2.1.2 Classification supervisée

Dans le cadre de ce projet, nous entraînons des algorithmes à classifier des cris selon l'état de santé du nourrisson dont ils proviennent. Les classifieurs sont des algorithmes qui analysent des échantillons, ou observations, pour tenter de déterminer à quelle classe ils appartiennent.

De plus, dans le cadre de ce projet, les classifieurs sont entraînés de façon supervisée. Lors de l'entraînement supervisé d'un classifieur, on se base sur des observations dont la classe est connue, pour apprendre à classifier correctement des observations dont la classe n'est pas connue.

2.2.2 Données

L'apprentissage machine est basé sur les données. Toutes les relations apprises par les algorithmes d'apprentissage découlent des données utilisées lors de l'entraînement.

Des données annotées sont requises pour réaliser ce projet, car nous entraînons des classifieurs de façon supervisée. Cela signifie que la classe réelle de chacun des observations utilisés lors de l'entraînement doit être connue.

La base de données utilisée dans ce projet est décrite à la section 3.1

2.2.3 Modèles

2.2.3.1 Réseaux de neurones

Les réseaux de neurones sont des modèles connexionnistes, c'est-à-dire qu'ils réalisent une fonction complexe en assemblant plusieurs fonctions simples.

Les réseaux de neurones sont généralements construits en connectant plusieurs couches l'une à la suite de l'autre. Les couches sont constituées de plusieurs opérations. En général, chaque couche contient une opération linéaire, sur laquelle est appliquée une fonction non-linéaire, aussi connue sous le nom de fonction d'activation.

La dernière couche est appelée couche de sortie. Les autres couches sont appelées couches cachées.

On peut voir un réseau de neurones comme un algorithme d'extraction de *features* guidé par les données. En effet, les couches cachées d'un classifieur à base de réseau de neurones apprennent à extraire des *features* pertinents à la tâche, tandis que la couche de sortie apprend à classifier correctement ces *features* (Goodfellow *et al.*, 2016).

2.2.3.2 Réseaux MLP

Les réseaux de neurones MLP (*Multi Layer Perceptron*) sont les plus simples. Ils sont constitués de plusieurs couches entièrement connectées, sans aucune rétroaction. Une simple transformation affine sert d'opération linéaire pour chaque couche.

Ce type de réseau permet d'approximer une fonction f^* décrivant un phénomène. Par exemple, dans le cadre de ce projet de classification supervisée, on assume qu'il existe une fonction f^* qui décrit le phénomène reliant toute observation $\boldsymbol{x}$ à une classe y, tel que $y = f^*(\boldsymbol{x})$.

Le réseau MLP définit une fonction $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ et se base sur un échantillonnage de $\boldsymbol{x}$ pour apprendre les paramètres $\boldsymbol{\theta}$ qui produisent la meilleure approximation possible de f^* (Goodfellow *et al.*, 2016).

Chaque couche ℓ commence par calculer la transformation linéaire $\boldsymbol{z}^{(\ell)}$ de son entrée $\boldsymbol{h}^{(\ell-1)}$

$$z_i^{(\ell)} = \left(\sum_j W_{ij}^{(\ell)} h_j^{(\ell-1)} \right) + b_i^{(\ell)} \tag{2.8}$$

ou, sous forme matricielle :

$$\boldsymbol{z}^{(\ell)} = \boldsymbol{W}^{(\ell)} \boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)} \tag{2.9}$$

Où $\boldsymbol{W}^{(\ell)}$ et $\boldsymbol{b}^{(\ell)}$ sont la matrice de poids et le vecteur de biais de la couche, respectivement.

La première couche reçoit l'observation $\boldsymbol{x}$ en entrée, c'est-à-dire $\boldsymbol{h}^{(0)} = \boldsymbol{x}$.

La couche applique ensuite une fonction non-linéaire $a^{(\ell)}$ sur la transformation linéaire $\boldsymbol{z}^{(\ell)}$.

$$\boldsymbol{h}^{(\ell)} = a^{(\ell)} \left(\boldsymbol{z}^{(\ell)} \right) \tag{2.10}$$

Les fonctions non-linéaires sont aussi appelées fonctions d'activation. Différentes fonctions d'activation peuvent être utilisées. Dans le cadre de ce projet, nous utilisons l'activation *ReLU* (*Rectified Linear Unit*). La fonction est appliquée séparément sur chaque élément $z_i^{(\ell)}$ du vecteur $\boldsymbol{z}^{(\ell)}$.

$$h_i^{(\ell)} = \text{ReLU} \left(z_i^{(\ell)} \right) = \max \left\{ 0,\ z_i^{(\ell)} \right\} \tag{2.11}$$

Dans les tâches de classification supervisée, la sortie du réseau doit retourner la classe prédite pour l'observation en entrée. Cela peut être réalisé de plusieurs manières. Dans le cadre de ce projet, le réseau de L couches produit un vecteur $\hat{\boldsymbol{y}} = \boldsymbol{h}^{(L)}$, où la valeur de chaque élément $\hat{y}_i$ correspond à la probabilité que l'observation appartienne à la classe i. L'activation *softmax* sert donc de fonction non-linéaire sur la couche de sortie, afin de produire des probabilités normalisées.

$$\hat{y}_i = h_i^{(L)} = \text{softmax}_i\left(\boldsymbol{z}^{(L)}\right) \tag{2.12}$$

$$= \frac{\exp\left(z_i^{(L)}\right)}{\sum_j \exp\left(z_j^{(L)}\right)} \tag{2.13}$$

Un exemple de réseau MLP respectant l'architecture utilisée dans le cadre ce projet est présenté à la figure 2.2

Le théorème d'approximation universelle (Cybenko, 1989), (Hornik, 1991) stipule qu'un réseau MLP d'une seule couche cachée permet d'approximer n'importe qu'elle fonction, avec une précision arbitraire, tant que le nombre de neurones sur la couche est suffisant et que les fonction d'activations de la couche cachée respectent certaines conditions (Hornik, 1991). Mentionnons néanmoins qu'il n'est pas garanti que l'algorithme d'entraînement parvienne à apprendre cette fonction.

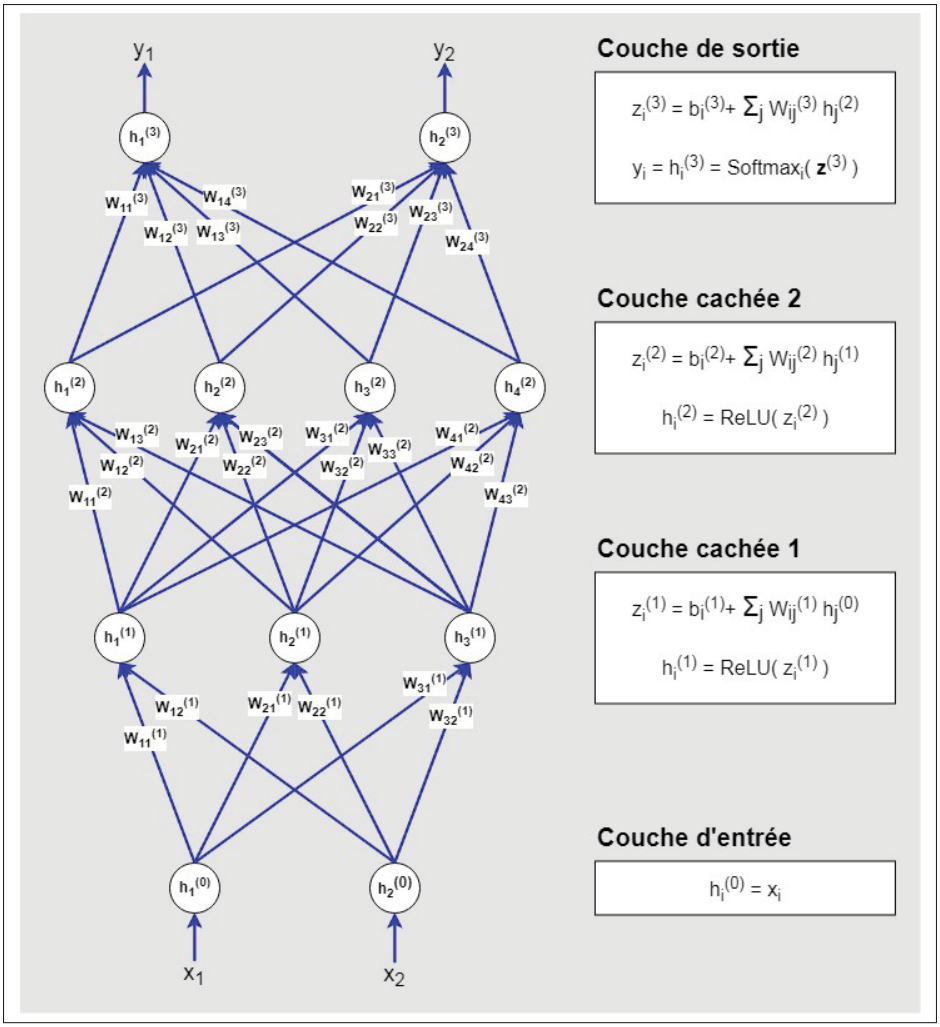

Figure 2.2 Exemple de réseau MLP à 2 couches cachées de 3 et 4 neurones, respectant l'architecture utilisée dans ce projet

2.2.3.3 Réseaux convolutionnnels

Les réseaux convolutionnels sont des réseaux spécialisés dans le traitement de signaux organisés en grille, comme les images, qui peuvent être vues comme une grille 2D de pixels, ou les séries temporelles, qui peuvent être vues comme une grille 1D de mesures (Goodfellow *et al.*, 2016).

Les réseaux convolutionnels sont des réseaux qui se basent sur la convolution comme opération linéaire, plutôt que la transformation affine.

La convolution est une opération similaire à la corrélation croisée. En effet, la convolution du tenseur $\boldsymbol{X}$ par le tenseur $\boldsymbol{W}$ est égale à la corrélation croisée du tenseur $\boldsymbol{X}$ par le mirroir du tenseur $\boldsymbol{W}$.

La convolution peut-donc être vue comme une mesure de corrélation. Elle mesure la ressemblance entre deux tenseurs, pour différents décalages. Un exemple de convolution est présenté à la figure 2.3.

Les couches convolutionnelles utilisent la convolution entre le tenseur d'entrée $\boldsymbol{X}$ et un tenseur $\boldsymbol{W}$, qu'on appelle *kernel*, comme opération linéaire Le résultat de la convolution est appelé *feature map*.

Le *feature map* est fortement activé aux positions où le tenseur d'entrée ressemble au *kernel*. La convolution permet donc de chercher, dans le tenseur d'entrée, le motif représenté par le *kernel*. Cette opération est répétée pour plusieurs *kernels*, afin de détecter différents motifs. Les *kernels* font partie des paramètres du réseau et sont donc appris lors de l'entraînement (Goodfellow *et al.*, 2016).

L'utilisation de la convolution permet de tirer avantage de trois principes : Les interactions clairsemées (*sparse*), le partage de paramètres et l'équivariance aux translations (Goodfellow *et al.*, 2016).

Premièrement, alors que dans les MLP la valeur d'un neurone dépend de tous les neurones des couches précédentes, dans les réseaux convolutionnels elle ne dépend que de quelques neurones de la couche précédente. On dit donc que les interactions dans les réseaux convolutionnels sont clairsemées. Cette propriété provient de la petite taille des *kernels*.

Deuxièmement, la convolution force le partage de paramètres. En effet, dans les réseaux convolutionnels, les paramètres, qui forment les *kernels*, sont réutilisés pour chaque décalage de l'entrée. Autrement dit, plutôt que d'apprendre un ensemble de paramètres propres à chaque position, on apprend un seul ensemble de paramètres qui décrit toutes les positions.

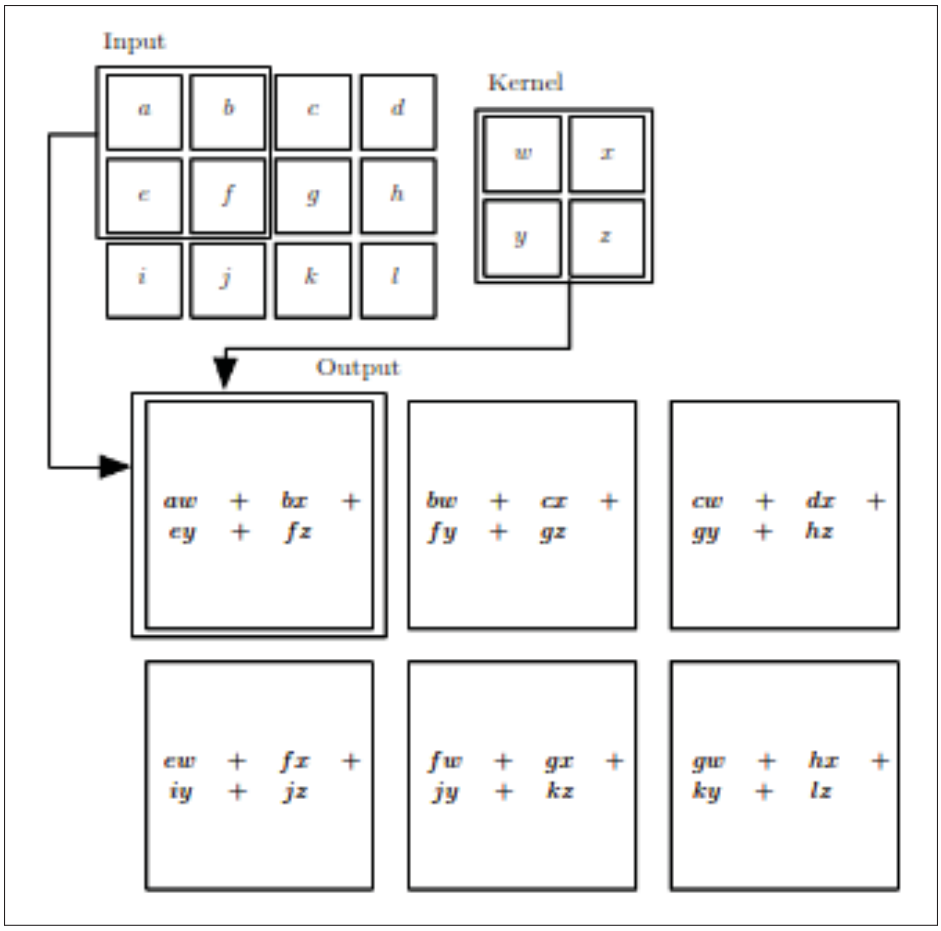

Figure 2.3 Exemple de convolution 2D,
tiré de Goodfellow *et al.* (2016, p.330)

Troisièmement, la convolution est une opération équivariante à la translation. En effet, la translation de la convolution d'un signal est équivalente à la convolution du signal translaté par la même quantité. Autrement dit, si la fonction $C(\boldsymbol{x}, \boldsymbol{w})$ représente la convolution de $\boldsymbol{x}$ par $\boldsymbol{w}$ et la fonction $f(\boldsymbol{x})$ effectue une translation de $\boldsymbol{x}$, alors $C\left(f\left(\boldsymbol{x}\right), \boldsymbol{w}\right) = f\left(C\left(\boldsymbol{x}, \boldsymbol{w}\right)\right)$. Cette opération n'est cependant pas équivariante à d'autres transformations comme la rotation ou le changement d'échelle

Ces propriétés apportent plusieurs avantages pour le traitement de données structurées en grille. Par exemple, cette formulation force le réseau à apprendre à reconnaître des motifs locaux, invariants aux translations. De plus, les données sont utilisées de façon plus efficace. En effet,

un ensemble de paramètres chargé de reconnaître un motif particulier apprendra de n'importe quelle image contenant le motif, peu importe sa position. Aussi, les interactions clairsemées réduisent de beaucoup la quantité de calcul et de mémoire requis pour entraîner le réseau.

Les hyperparamètres de chaque convolution sont le nombre *kernels* et leur taille dans chacune des dimensions. De plus, il est possible d'utiliser un décalage de plus d'un, ce qui est équivalent à sous-échantillonner les *feature maps* produits. Les incréments des décalages dans chacune des dimensions s'ajoutent donc aux hyperparamètres. Aussi, la façon de calculer de la convolution sur les extrémités de l'entrée peut également être ajustée selon le problème.

Une couche convolutionnelle typique est composée d'au moins deux fonctions. La première fonction calcule la convolution de l'entrée avec plusieurs *kernels* pour produire les *feature maps*. La seconde fonction, la fonction d'activation, applique une transformation non-linéaire sur les *feature maps* pour produire les activations. Une troisième fonction, la fonction de *pooling*, suit souvent.

Les fonctions de *pooling* calculent des statistiques sommaires locales des activations. L'opération *max pooling*, par exemple, retourne les maximums locaux des activations. Les maximum locaux sont les maximums de zones espacées également. La taille des zones et l'espacement entre les zones sont tous deux des hyperparamètres de la fonction de *pooling*.

Le *pooling* permet de rendre la couche plus invariante aux translations. Il est utile lorsqu'on s'intéresse à la présence ou l'absence d'un motif, sans avoir besoin de connaître sa position exacte (Goodfellow *et al.*, 2016)

Les réseaux convolutionnels sont spécialisés dans le traitement de signaux structurés en grille. Pour s'en convaincre, comparons deux réseaux convolutionnels. Le premier est entraîné avec des images normales. Le second est entraîné avec les mêmes images, mais dont les pixels ont été permutés. Le second réseau obtiendra de bien moins bonnes performances que le premier, car la structure des signaux traités a été détruite. Un MLP, à l'inverse, ne verrait aucune différence.

Les réseaux convolutionnels ont principalement étés inventés pour le traitement d'images, qui ont une structure spatiale. Les signaux ayant une structure séquentielle, comme les signaux audio, ou les représentations temps-fréquence, comme les spectrogrammes, ont également avantage à être traités avec des réseaux convolutionnels.

Les réseaux convolutionnels ont permis de grandes avancées dans le domaine de la reconnaissance d'images. L'exemple le plus notoire est celui de la compétition ILSVRC (*ImageNet large scale visual recognition challenge*), qui porte sur la reconnaissance de centaines de catégories d'objets dans des millions d'images (Russakovsky *et al.*, 2015). Les compétitions furent toutes remportées par des réseaux convolutionnels à partir de l'an 2012, où le réseau *Alexnet* (Krizhevsky *et al.*, 2012) remporta la compétition avec un *top-five error rate* de 17.0%, alors que le meilleur jamais obtenu était de 25.7%. Ce chiffre diminua radicalement au cours des années suivantes, pour atteindre 2.251% en 2017 (Hu *et al.*, 2017).

Dans le cadre de notre projet, nous avons créé un réseau convolutionnel, dont chaque couche cachée était constituée d'une convolution 2D suivie d'un *max pooling* puis d'une activation *ReLU*. La sortie de la dernière couche convolutionnelle était réarrangée en un vecteur, qui servait d'entrée à la couche de sortie. La couche de sortie calculait la transformation affine, suivie de l'activation *softmax* afin de produire le vecteur contenant la probabilité de chacune des classes.

2.2.3.4 Réseaux récurrents et cellules LSTM

Les réseaux récurrents sont des réseaux spécialisés dans le traitement de séquences, grâce à l'incorporation de rétroactions dans le modèle. Ils tirent avantage du principe de partage de paramètres, de façon similaire aux réseaux convolutionnels.

Comme l'explique Goodfellow *et al.* (2016), si on créait un MLP pour traiter des phrases de longueur fixe, le réseau utiliserait des paramètres différents sur chaque élément de la séquence, ce qui le forcerait à apprendre les règles du langage séparément pour chaque position dans la séquence. À l'inverse, dans un réseau récurrent, les mêmes paramètres servent à modéliser

chaque position dans la séquence, les règles du langage ne sont donc apprises qu'une seule fois. De plus, cela permet au réseau de traiter des séquences de longueur arbitraite.

On veut constuire une fonction ϕ qui produit à chaque instant t une sortie $\boldsymbol{o}^{(t)}$ dépendante de la séquence d'entrées observée jusqu'à présent $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$.

$$\boldsymbol{o}^{(t)} = \phi\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}\right) \tag{2.14}$$

Comment implémenter le modèle d'une telle fonction ? Plus la séquence est longue, plus la fonction dépend d'un grand nombre d'entrées.

Un réseau récurrent implémente cette fonction en introduisant une variable d'état $\boldsymbol{h}^{(t)}$. Dans la majorité des réseaux récurrents, l'état à chaque instant t dépend de l'observation actuelle et de l'état précédent.

$$\boldsymbol{h}^{(t)} = f\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}_f\right) \tag{2.15}$$

La sortie à chaque instant t est ensuite calculée à partir de la variable d'état.

$$\boldsymbol{o}^{(t)} = g\left(\boldsymbol{h}^{(t)}; \boldsymbol{\theta}_g\right) \tag{2.16}$$

La variable d'état contient donc l'information pertinente des observations précédentes. L'entraînement ajuste les paramètres $\boldsymbol{\theta}_f$ de la fonction f de manière à extraire ces informations pertinentes. Il ajuste également les paramètres $\boldsymbol{\theta}_g$ de la fonction g de sorte à produire les bonnes sorties à partir de ces informations. Cette formulation permet d'utiliser les mêmes paramètres à toutes les positions dans la séquence. Elle permet également de modéliser des séquences de longueur variable.

Les fonctions f et g sont typiquement réalisées par une transformation affine suivie d'une non-linéarité, comme dans les MLP.

Cette formulation apporte cependant certains problèmes. En effet, grâce au partage de paramètres, les relations à court et à long terme sont modélisées par les mêmes paramètres. Cependant, lors de l'apprentissage, les relations à court terme influençent le gradient de façon exponentiellement plus élevée que les relations à long terme. Or, les algorithmes d'entraînement des paramètres sont basés sur le gradient (voir section 2.2.5.1). Le réseau apprend donc beaucoup plus facilement à modéliser les relations à court terme que les relations à long terme. De plus, le bruit des gradients des relations à court terme vient masquer les gradients des relations à long terme, limitant l'apprentissage des relations à long terme. Il s'agit du problème des gradients disparaissants (*vanishing gradients*)

La cellule LSTM (*long short-term memory*) a été créée pour tenter de résoudre cette difficulté. Dans la cellule LSTM, chaque élément de la sortie $\boldsymbol{h}^{(t)}$ est fonction de l'élément correspondant dans la variable d'état $\boldsymbol{s}^{(t)}$.

$$h_i^{(t)} = q_i^{(t)} \tanh\left(s_i^{(t)}\right) \tag{2.17}$$

La fonction tanh() sert de fonction d'activation. Le vecteur de poids $\boldsymbol{q}^{(t)}$ est appelé *output gate*. Il contrôle la proportion du signal d'état qui est propagé à la sortie.

Chaque élément de la variable d'état $\boldsymbol{s}^{(t)}$ est fonction de sa valeur à l'instant précédent, ainsi que de l'élément correspondant dans le vecteur $\boldsymbol{v}^{(t)}$.

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} v_i^{(t)} \tag{2.18}$$

Le premier terme implémente la rétroaction. Le second terme permet d'accumuler l'information de l'entrée. Le vecteur de poids $\boldsymbol{f}^{(t)}$ est appelé *forget gate*. Il permet d'ajuster la force de la rétroaction. Le vecteur de poids $\boldsymbol{g}^{(t)}$ est appelé *input gate*. Il permet d'ajuster la proportion d'accumulation des caractéristiques.

Le vecteur $\boldsymbol{v}^{(t)}$ est calculé à partir de l'entrée $\boldsymbol{x}^{(t)}$ et de la sortie $\boldsymbol{h}^{(t)}$ par une transformation affine suivie d'une fonction non-linéaire, comme la fonction sigmoïde ou la tangente hyperbolique.

Dans le cadre de ce projet, nous utilisons la tangente hyperbolique.

$$v_i^{(t)} = \tanh\left(b_i^v + \sum_j U_{ij}^v x_j^{(t)} + \sum_j W_{ij}^v h_j^{(t-1)}\right) \tag{2.19}$$

Où $\boldsymbol{U}^v$ et $\boldsymbol{W}^v$ sont les matrices de poids et $\boldsymbol{b}^v$ le vecteur de biais de la transformation affine utilisée lors du calcul du vecteur $\boldsymbol{v}$ pour chaque instant t.

Les *gates* $\boldsymbol{q}^{(t)}$, $\boldsymbol{f}^{(t)}$ et $\boldsymbol{g}^{(t)}$ sont calculées dynamiquement. Leur valeur change donc à chaque position dans la série. Elles sont toutes calculées à partir de l'entrée $\boldsymbol{x}^{(t)}$ et de la sortie $\boldsymbol{h}^{(t)}$ par une transformation affine suivie d'une fonction non-linéaire. La fonction sigmoïde $\sigma()$ est utilisée, afin de produire des valeurs entre 0 et 1.

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{ij}^g x_j^{(t)} + \sum_j W_{ij}^g h_j^{(t-1)}\right) \tag{2.20}$$

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{ij}^f x_j^{(t)} + \sum_j W_{ij}^f h_j^{(t-1)}\right) \tag{2.21}$$

$$q_i^{(t)} = \sigma\left(b_i^q + \sum_j U_{ij}^q x_j^{(t)} + \sum_j W_{ij}^q h_j^{(t-1)}\right) \tag{2.22}$$

La cellule a donc le contrôle sur la force des signaux d'entrée, de sortie et de rétroaction. Elle peut ajuster les poids des *gates* afin d'accumuler l'information utile, puis la discarter lorsqu'elle n'est plus nécessaire. Cela permet à la cellule de contrôler la proportion d'information provenant des divers instants précédents. Si une relation à long terme est particulièrement importante, le réseau ajustera ses poids afin d'éviter le problème des gradients disparaissants.

L'architecture de la cellule LSTM peut être visualisée dans la figure 2.4. Le *dropout* est expliqué à la section 2.2.6.5

Plusieurs études ont démontré que les cellules LSTM apprennent plus facilement les relations à long terme (Bengio *et al.*, 1994), (Hochreiter *et al.*, 2001). Elles ont permis d'améliorer l'état de l'art dans plusieurs tâches de traitement de séries (Graves, 2013), (Sutskever *et al.*, 2014).

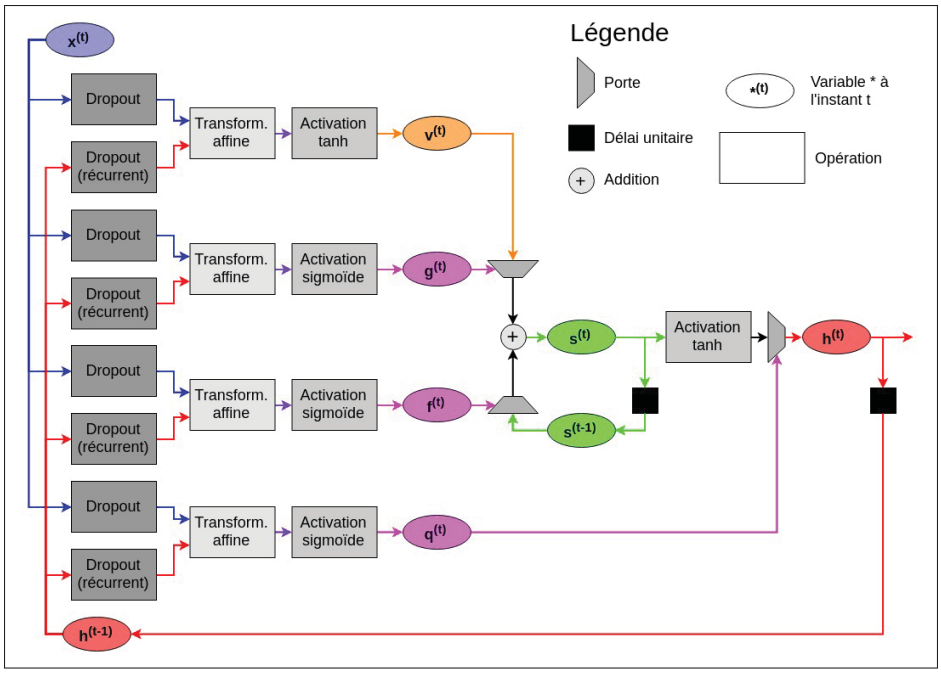

Figure 2.4 Diagramme du fonctionnement d'une cellule LSTM

Les séquences pourraient également être traitées avec des réseaux convolutionnels à une dimension. La durée des interactions serait cependant limitée par la taille des *kernels*.

Dans le cadre de ce projet, nous avons créé un réseau récurrent basé sur des cellules LSTM. Chaque couche cachée était constituée d'une cellule LSTM, prenant en entrée une séquence $\boldsymbol{h}^{(1)(\ell)}, \boldsymbol{h}^{(2)(\ell)}, \ldots, \boldsymbol{h}^{(\tau)(\ell)}$ et retournant en sortie une séquence $\boldsymbol{h}^{(1)(\ell+1)}, \boldsymbol{h}^{(2)(\ell+1)}, \ldots, \boldsymbol{h}^{(\tau)(\ell+1)}$.

La dernière couche cachée prenait une séquence en entrée mais ne retournait que le dernier élément de la séquence produite, $\boldsymbol{h}^{(\tau)(L-1)}$.

Ce dernier élément servait d'entrée à la couche de sortie, qui calculait une transformation affine $\boldsymbol{z}^{(L)} = \boldsymbol{W}^{(L)}\boldsymbol{h}^{(\tau)(L-1)} + \boldsymbol{b}^{(L)}$ suivie d'une activation *softmax* afin de produire le vecteur de probabilité de chacune des classes, $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{z}^{(L)})$.

2.2.4 Fonction de coût

La fonction de coût mesure la qualité des paramètres du modèle. Elle est utilisée lors de l'entraînement pour guider l'ajustement des paramètres.

En entraînement supervisé, le coût peut être constitué de la somme de plusieurs termes, dont le plus important est la perte. La perte mesure la différence entre les valeurs prédites par le modèle et les valeurs réelles. Plus la différence est importante, plus la perte, et donc le coût, augmentent. L'objectif de l'entraînement est donc de diminuer le coût retourné par le modèle pour les données d'entraînement. Le coût contient souvent d'autres termes, qui permettent de favoriser certaines solutions par rapport à d'autres. Un terme de régularisation, par exemple, est souvent ajouté (pour plus de détails, voir la section 2.2.6.2).

Les fonctions de perte sont souvent basées sur le principe du maximum de vraisemblance *maximum likelihood*). Dans le cas de l'entraînement supervisé, ce principe se traduit par la minimisation de l'entropie croisée entre la distribution empirique des données et la distribution estimée par le modèle. L'entropie croisée de ces distributions sert donc de fonction de perte (Goodfellow *et al.*, 2016).

Dans la cadre de ce projet, nous entraînons des classifieurs de façon supervisée. Prenons une tâche de classification à n classes. Chacun de nos réseaux retourne un vecteur $\hat{\boldsymbol{y}}$. La valeur de chacun des éléments $\hat{y}_i$ du vecteur correspond à la probabilité estimée par le modèle que l'observation $\boldsymbol{x}$ appartienne à la classe i.

$$\hat{y}_i = p(i|\boldsymbol{x}, \boldsymbol{\Theta}), \quad 1 \leq i \leq n \tag{2.23}$$

Définissons également le vecteur $\boldsymbol{y}$, qui est l'encodage *one-hot* de la classe réelle de l'observation, c. Ce vecteur est nul, excepté à l'index correspondant à la classe de l'observation, où il vaut 1. Autrement dit, si l'observation appartient à la classe i, alors chaque élément y_j du

vecteur vaut :

$$y_j = \begin{cases} 1 & j = i \\ & \\ 0 & j \neq i \end{cases}, \quad 1 \leq i \leq n \tag{2.24}$$

La fonction de perte, constituée de l'entropie croisée entre les prédictions $\hat{\boldsymbol{y}}$ et la réalité $\boldsymbol{y}$ est :

$$\mathscr{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{i=1}^{n} y_i \log \hat{y}_i \tag{2.25}$$

Dans le cadre de ce projet, la fonction de coût est uniquement consituée de la perte.

$$J(\boldsymbol{\theta}) = \mathscr{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \tag{2.26}$$

2.2.5 Entraînement et optimisation

L'entraînement supervisé consiste à ajuster les paramètres d'un modèle en se basant sur des données annotées. Les paramètres sont ajustés de sorte que le modèle prédise le mieux possible l'étiquette de ces données. Dans le cadre de ce projet, on ajuste les paramètres de réseaux de neurones en se basant sur des observations provenant d'enregistrements de cris de nourrissons, dont l'état de santé est connu. Les paramètres sont ajustés de sorte que les réseaux prédisent le mieux possible l'état de santé des nourrissons dont proviennent les observations.

L'ajustement des paramètres se pose comme un problème d'optimisation. On mesure la qualité des paramètres à l'aide de la fonction de coût (voir section 2.2.4). On optimise ensuite les paramètres en minimisant cette fonction de coût.

Dans la majorité des cas, il est impossible de trouver les paramètres optimaux de façon analytique. Il faut alors avoir recours à une méthode itérative.

2.2.5.1 Descente du gradient

Les algorithmes de descente du gradient se servent du gradient de la fonction de coût pour ajuster itérativement les paramètres du modèle de façon à minimiser le coût.

En effet, le gradient de la fonction de coût par rapport aux paramètres pointe dans la direction qui fait augmenter le coût le plus rapidement. L'inverse du gradient indique donc la direction qui fait diminuer le coût le plus rapidement. La descente du gradient se sert de cette information pour guider l'ajustement des paramètres à chaque itération.

Le gradient du coût de la i$^{\text{ième}}$ l'observation, $\nabla J_i(\boldsymbol{\theta})$, peut être obtenu grâce à l'algorithme de rétropropagation, présenté à la section 2.2.5.2.

Les algorithmes de descente du gradient par *batch* (*batch gradient descent*) se servent à chaque itération de la moyenne des gradients de toutes les données d'entraînement pour effectuer un ajustement des paramètres. Sous sa forme la plus simple, la descente du gradient par *batch* est décrite par l'équation suivante :

$$\theta_{k+1} \leftarrow \theta_k - \alpha_k \frac{1}{m} \sum_{i=1}^{m} \nabla J_i(\boldsymbol{\theta}_k) \tag{2.27}$$

où m est le nombre d'observations, k est l'indice de l'itération et α_k est le taux d'apprentissage, qui permet de contrôler la taille de l'ajustement.

Étant donné que le gradient exact des données d'entraînement est calculé, chaque itération produira une amélioration fiable des paramètres du modèle. Chaque itération de la descente du gradient par *batch* est cependant très coûteuse en ressources de calcul, car le gradient de toutes les observations doit être évalué. Le calcul des gradients est heureusement facilement parrallélisable (Bottou *et al.*, 2018).

Les algorithmes de descente stochastique du gradient, quant à eux, se servent, à chaque itération, du gradient d'un seul exemple d'entraînement, sélectionné aléatoirement, pour effectuer

un ajustement des paramètres. Sous sa forme la plus simple, la descente stochastique du gradient est décrite par l'équation suivante :

$$\theta_{k+1} \leftarrow \theta_k - \alpha_k \nabla J_{i_k}(\boldsymbol{\theta}_k) \tag{2.28}$$

où k est l'indice de l'itération et i_k est l'indice de l'exemple sélectionné aléatoirement à l'itération k.

Chaque itération de la descente stochastique du gradient est très peu coûteuse, car un seul gradient est calculé. La valeur calculée est cependant une estimation très peu fiable du gradient de toutes les données d'entraînement. On s'attend donc à ce qu'en moyenne chaque itération apporte une moins bonne diminution du coût.

La descente stochastique du gradient a cependant l'avantage d'utiliser plus efficacement l'information. Goodfellow *et al.* (2016) donnent comme exemple un modèle entraîné avec m observations identiques. La descente stochastique du gradient calculerait exactement le même gradient que la descente du gradient par *batch*, mais requièrait m fois moins de calculs. Bien qu'on ne retrouve pas ce cas extrême en pratique, en général les données d'entraînement contiennent beaucoup d'observations similaires.

Les algorithmes de descente du gradient par *mini-batch* cherchent un compromis entre la descente par *batch* et la descente stochastique. Ces algorithmes se servent à chaque itération de la moyenne du gradient de plusieurs observations sélectionnés aléatoirement pour effectuer un ajustement des paramètres. Le sous-ensemble d'observations sélectionnées aléatoirement à chaque itération est appelé *mini-batch*. Le nombre d'observations dans le *mini-batch* est un hyperparamètre. Sous sa forme la plus simple, la descente du gradient par *mini-batch* est décrite par l'équation suivante :

$$\theta_{k+1} \leftarrow \theta_k - \alpha_k \frac{1}{m_k} \sum_{i \in S_k} \nabla J_i(\boldsymbol{\theta}_k) \tag{2.29}$$

où S_k est l'ensemble des indices des observations du *mini-batch* et m_k est la taille du *mini-batch*, à l'itération k. Cette technique de descente du gradient combine les forces de la descente par *batch* et la descente stochastique (Bottou *et al.*, 2018).

Le taux d'apprentissage α_k est un hyperparamètre important. En général, on diminue le taux d'apprentissage au fur et à mesure que l'entraînement progresse, afin de se rapprocher le plus possible du minimum. L'ajustement du taux d'apprentissage peut être effectué manuellement, mais il s'agit d'une tâche difficile. Des algorithmes calculant les taux d'apprentissage de façon adaptative ont donc été créés. Ces algorithmes ont également l'avantage d'utiliser un taux d'apprentissage différent pour chacun des paramètres.

La méthode de la quantité de mouvement (*momentum*), introduite par Polyak (1964), est une technique permettant d'accélérer la descente du gradient. Cet algorithme se déplace sur la surface de coût "en conservant son élan", à la manière d'une balle descendant une colline. Avec cette technique, la direction de l'ajustement des paramètres dépend non-seulement de la direction du gradient, mais aussi de la direction des gradients précédents. De même, la taille de l'ajustement dépend non-seulement du gradient, mais aussi de la norme et de l'alignement des gradients précédents (Goodfellow *et al.*, 2016). Cette modification permet d'accélérer l'apprentissage, particulièrement lorsque la surface de coût présente une grande courbure, ou lorsque les gradients sont bruités, ou lorsque les gradients sont petits mais inchangeants.

Dans le cadre de ce projet, nous utilisons la méthode Adam (Kingma & Ba, 2015). Cette méthode combine les techniques de *momentum* et de calcul adaptatif de taux d'apprentissage propres à chaque paramètre (Goodfellow *et al.*, 2016).

2.2.5.2 Rétropropagation

La rétropropagation (Rumelhart *et al.*, 1986) est un algorithme qui permet de calculer les gradients d'une composition de fonctions paramétrisées de façon efficace, basé sur la règle de chaîne (Goodfellow *et al.*, 2016).

Le gradient d'une fonction scalaire $f(\boldsymbol{x})$ par rapport à $\boldsymbol{x}$ est un vecteur contenant les dérivées partielles de la fonction par rapport à chacune des dimensions de $\boldsymbol{x}$.

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2} \\ \vdots \end{bmatrix} \tag{2.30}$$

La règle de chaîne permet de décomposer le calcul du gradient de fonctions composées.

Supposez $c = g(\boldsymbol{b})$ et $\boldsymbol{b} = h(\boldsymbol{a})$, tel que $c = g\left(h(\boldsymbol{a})\right)$. La variable c est produite par la composition des fonctions h et g. Le calcul de la dérivée partielle de c par rapport à n'importe quel élément a_i du vecteur $\boldsymbol{a}$ peut être décomposé ainsi :

$$\frac{\partial c}{\partial a_i} = \sum_j \frac{\partial c}{\partial b_j} \frac{\partial b_j}{\partial a_i} \tag{2.31}$$

ce qui est équivalent à la forme matricielle :

$$\nabla_{\boldsymbol{a}} c = \left(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}\right)^{\mathsf{T}} \nabla_{\boldsymbol{b}} c \tag{2.32}$$

où $\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}}$ est une matrice jacobienne. La matrice jacobienne de la fonction $\boldsymbol{b} = h(\boldsymbol{a})$ contient les dérivées partielles de tous les b_i par rapport à tous les a_j.

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{a}} = \begin{bmatrix} \frac{\partial b_1}{\partial a_1} & \frac{\partial b_1}{\partial a_2} & \cdots \\ \frac{\partial b_2}{\partial a_1} & \frac{\partial b_2}{\partial a_2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{2.33}$$

En général, les réseaux de neurones sont composés de couches, qui sont des composées d'opérations paramétrisées. Les réseaux de neurones peuvent donc être décrits comme une composi-

tion de fonctions différentiables paramétrisées. Une des motivations pour construire les réseaux de cette façon est de pouvoir simplifier le calcul des gradients, grâce à la règle de chaîne.

La rétroprogation est un algorithme basé sur la règle de chaîne qui permet de calculer les gradients de façon efficace, en prenant avantage du fait que plusieurs termes apparaissant dans le calcul des gradients par rapport aux paramètres des couches supérieures se répètent dans le calcul des gradients par rapport aux paramètres des couches inférieures.

Prenons par exemple un simple réseau MLP de $L = 3$ couches cachées, produisant un vecteur de probabilités normalisées $\hat{\boldsymbol{y}}$. Chaque couche ℓ applique une fonction d'activation $\boldsymbol{h}^{(\ell)} = a^{(\ell)}(\boldsymbol{z}^{(\ell)})$ sur la transformation affine $\boldsymbol{z}^{(\ell)} = f(\boldsymbol{h}^{(\ell-1)}; \boldsymbol{\theta}^{(\ell)}) = \boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}$. La fonction *softmax* sert d'activation pour la couche de sortie. Pour ce qui est des couches cachées, utilisons des activations *ReLU*.

Les paramètres de chaque couche ℓ sont contenus dans le vecteur $\boldsymbol{\theta}^{(\ell)}$ et l'ensemble des paramètres de toutes les couches est contenu dans le vecteur $\boldsymbol{\Theta}$

$$\boldsymbol{\theta}^{(\ell)} = \begin{bmatrix} w_{11}^{(\ell)} & w_{12}^{(\ell)} & \dots & b_1^{(\ell)} & b_2^{(\ell)} & \dots \end{bmatrix} \tag{2.34}$$

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}^{(1)} & \boldsymbol{\theta}^{(2)} & \boldsymbol{\theta}^{(3)} \end{bmatrix} \tag{2.35}$$

La variable $\boldsymbol{h}^{(0)}$ est égale au vecteur d'entrée $\boldsymbol{x}$, si bien que le calcul effectué par un réseau de $L = 3$ couches est :

$$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(3)} = a^{(3)}\left(f\left(a^{(2)}\left(f\left(a^{(1)}\left(f\left(\boldsymbol{x}; \boldsymbol{\theta}^{(1)}\right)\right); \boldsymbol{\theta}^{(2)}\right)\right); \boldsymbol{\theta}^{(3)}\right)\right) \tag{2.36}$$

Utilisons un coût constitué uniquement de la perte , qui est l'entropie croisée entre $\hat{\boldsymbol{y}}$, le vecteur de probabilités pour chaque classe estimé par le réseau, et $\boldsymbol{y}$, l'encodage *one-hot* de la classe réelle de l'observation.

$$J(\boldsymbol{\Theta}) = \mathscr{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) - \sum_i y_i \log \hat{y}_i \tag{2.37}$$

On cherche à calculer le gradient du coût de l'observation i, $J_i(\boldsymbol{\Theta})$ en fonction de l'ensemble des paramètres $\boldsymbol{\Theta}$. Ce gradient est constitué de la concaténation des gradients du coût $J(\boldsymbol{\Theta})$ par rapport aux paramètres $\boldsymbol{\theta}^{(\ell)}$ de chaque couche.

$$\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}) = \begin{bmatrix} \frac{\partial J(\boldsymbol{\Theta})}{\partial \Theta_1} \\ \frac{\partial J(\boldsymbol{\Theta})}{\partial \Theta_2} \\ \vdots \end{bmatrix} = \begin{bmatrix} \nabla_{\boldsymbol{\theta}^{(1)}} J(\boldsymbol{\Theta}) \\ \nabla_{\boldsymbol{\theta}^{(2)}} J(\boldsymbol{\Theta}) \\ \vdots \end{bmatrix} \tag{2.38}$$

On peut utiliser la règle de chaîne pour décomposer les gradients $\nabla_{\boldsymbol{\theta}^{(\ell)}} J(\boldsymbol{\Theta})$.

$$\nabla_{\boldsymbol{\theta}^{(1)}} J(\boldsymbol{\Theta}) = \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}^{(3)}} \frac{\partial \boldsymbol{z}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{z}^{(2)}} \frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{z}^{(1)}} \frac{\partial \boldsymbol{z}^{(1)}}{\partial \boldsymbol{\theta}^{(1)}} \right)^\intercal \nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta}) \tag{2.39}$$

$$= \left(\frac{\partial \boldsymbol{z}^{(1)}}{\partial \boldsymbol{\theta}^{(1)}} \right)^\intercal \left(\frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{z}^{(1)}} \right)^\intercal \left(\frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \right)^\intercal \left(\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{z}^{(2)}} \right)^\intercal \left(\frac{\partial \boldsymbol{z}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \right)^\intercal \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}^{(3)}} \right)^\intercal \nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta}) \tag{2.40}$$

$$\nabla_{\boldsymbol{\theta}^{(2)}} J(\boldsymbol{\Theta}) = \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}^{(3)}} \frac{\partial \boldsymbol{z}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{z}^{(2)}} \frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{\theta}^{(2)}} \right)^\intercal \nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta}) \tag{2.41}$$

$$= \left(\frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{\theta}^{(2)}} \right)^\intercal \left(\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{z}^{(2)}} \right)^\intercal \left(\frac{\partial \boldsymbol{z}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \right)^\intercal \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}^{(3)}} \right)^\intercal \nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta}) \tag{2.42}$$

$$\nabla_{\boldsymbol{\theta}^{(3)}} J(\boldsymbol{\Theta}) = \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}^{(3)}} \frac{\partial \boldsymbol{z}^{(3)}}{\partial \boldsymbol{\theta}^{(3)}} \right)^\intercal \nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta}) \tag{2.43}$$

$$= \left(\frac{\partial \boldsymbol{z}^{(3)}}{\partial \boldsymbol{\theta}^{(3)}} \right)^\intercal \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}^{(3)}} \right)^\intercal \nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta}) \tag{2.44}$$

L'algorithme de rétropropagation profite du fait que plusieurs des termes calculés pour obtenir $\nabla_{\boldsymbol{\theta}^{(\ell)}} J(\boldsymbol{\Theta})$ sont réutilisés pour calculer $\nabla_{\boldsymbol{\theta}^{(\ell-1)}} J(\boldsymbol{\Theta})$.

La propagation avant pour cet exemple est présentée dans l'algorithme 2.1. Le calcul du coût et la rétroprogation pour cet exemple sont présentés dans l'algorithme 2.2.

Algorithme 2.1 Propagation avant

input : Observation $\boldsymbol{x}$ et paramètres du réseau $\boldsymbol{\Theta}$
output : Vecteur de probabilités estimées de chaque classe $\hat{\boldsymbol{y}}$

`/* Propagation avant, calcule les probabilités estimées par le réseau,` $\hat{\boldsymbol{y}}$ `*/`

$\boldsymbol{h}^{(0)} \leftarrow \boldsymbol{x}$
for $\ell \leftarrow 1$ **to** L **do**
  $\boldsymbol{z}^{(\ell)} = \boldsymbol{W}^{(\ell)} \cdot \boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}$
  $\boldsymbol{h}^{(\ell)} = a^{(\ell)}\left(\boldsymbol{z}^{(\ell)}\right)$
end
$\hat{\boldsymbol{y}} \leftarrow \boldsymbol{h}^{(L)}$

Algorithme 2.2 Calcul du coût et rétropropagation

input : Classe réelle de l'observation $\boldsymbol{y}$ et paramètres du réseau $\boldsymbol{\Theta}$
output : Gradient du coût de l'observation par rapport aux paramètres du réseau $\nabla_{\boldsymbol{\Theta}} J$

`/* Calcul du coût, mesure la différence entre la prédiction du réseau` $\hat{\boldsymbol{y}}$ `et la réalité` $\boldsymbol{y}$ `*/`

$J \leftarrow \mathscr{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$

`/* Rétropropagation, calcule le gradient du coût de l'observation par rapport aux paramètres,` $\boldsymbol{\Theta}$ `*/`

$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J$
for $\ell \leftarrow L$ **to** 1 **do**
  $\boldsymbol{g} \leftarrow \left(\frac{\partial \boldsymbol{h}^{(\ell)}}{\partial \boldsymbol{z}^{(\ell)}}\right)^{\mathsf{T}} \cdot \boldsymbol{g}$
  $\nabla_{\boldsymbol{\theta}^{(\ell)}} J(\boldsymbol{\Theta}) \leftarrow \left(\frac{\partial \boldsymbol{z}^{(\ell)}}{\partial \boldsymbol{\theta}^{(\ell)}}\right)^{\mathsf{T}} \cdot \boldsymbol{g}$
  $\boldsymbol{g} \leftarrow \left(\frac{\partial \boldsymbol{z}^{(\ell)}}{\partial \boldsymbol{h}^{(\ell-1)}}\right)^{\mathsf{T}} \cdot \boldsymbol{g}$
end
$\nabla_{\boldsymbol{\Theta}} J \leftarrow \left[\nabla_{\boldsymbol{\theta}^{(1)}} J,\ \nabla_{\boldsymbol{\theta}^{(2)}} J,\ \ldots,\ \nabla_{\boldsymbol{\theta}^{(L)}} J\right]$

Le calcul du gradient $\nabla_{\hat{\boldsymbol{y}}} J$ et des jacobiens $\frac{\partial \boldsymbol{h}^{(\ell)}}{\partial \boldsymbol{z}^{(\ell)}}$, $\frac{\partial \boldsymbol{z}^{(\ell)}}{\partial \boldsymbol{h}^{(\ell-1)}}$ et $\frac{\partial \boldsymbol{z}^{(\ell)}}{\partial \boldsymbol{\theta}^{(\ell)}}$ est présenté en annexe I.

Notez que le modèle MLP développé dans ce projet était légèrement plus complexe que cet exemple simple, chacune de ses couches contenant en plus une opération *dropout* et une opération *batch normalisation*.

2.2.5.3 Batch normalisation

Dans un réseau entraîné par descente du gradient (voir section 2.2.5.1), on ajuste les paramètres d'une couche en assumant que les autres couches restent inchangées. Cependant, en pratique, on ajuste les paramètres de toutes les couches en même temps. Cela peut produire des résultats inattendus, car les modifications aux couches inférieures affectent l'entrée reçue par les couches supérieures. Ce phénomène rend particulièrement plus difficile la sélection du taux d'apprentissage.

L'opération *batch normalisation* a été suggérée par Ioffe & Szegedy (2015) pour résoudre ce problème et faciliter l'entraînement des réseaux de neurones. La *batch normalisation* normalise les activations d'un *mini-batch* (voir section 2.2.5.1).

Plus précisément, définissons $\boldsymbol{H}$, une matrice contenant les activations d'une couche pour un *mini-batch* de m observations. Chaque rangée contient toutes les activations d'une des observations du *mini-batch*.

L'opération *batch normalisation* calcule $\boldsymbol{H}'$, une matrice contenant les activations normalisées du *mini-batch*.

$$\boldsymbol{H}' = \frac{\boldsymbol{H} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{2.45}$$

Où $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$ sont des vecteurs contenant la moyenne et l'écart-type des activations de chaque neurone pour le *mini-batch*.

$$\boldsymbol{\mu} = \frac{1}{m}\sum_i \boldsymbol{H}_{i,:} \tag{2.46}$$

$$\boldsymbol{\sigma} = \sqrt{\delta + \frac{1}{m}\sum_i (\boldsymbol{H} - \boldsymbol{\mu})_i^2} \tag{2.47}$$

avec une très petite valeur positive δ, ajoutée pour éviter les écarts-types nuls.

Lors de l'entraînement, les gradients sont propagés à travers le calcul de la moyenne, de l'écart-type et de la normalisation. L'opération de *batch normalisation* assure donc que la descente du gradient ne provoquera jamais de modification de la moyenne ou de l'écart-type de la sortie de la couche, facilitant donc l'apprentissage des couches suivantes.

Une couche linéaire ne peut qu'affecter les moments statistiques d'ordre 1 et 2. Son effet serait donc complètement annulé par l'opération *batch normalisation* (qui est également une opération linéaire).

Les couches non-linéaires, par contre, peuvent agir sur les moments statistiques d'ordre supérieur. L'application du *batch normalisation* permet donc de maintenir les moments statistiques d'ordre 1 et 2 (moyenne et variance) fixes, tout en permettant des changements aux moments statistiques d'ordre supérieur (*skewness*, *Kurtosis*, etc...).

La *batch normalisation* ne vient donc pas annuler l'effet des couches non-linéaires, elle permet simplement de faciliter l'apprentissage.

Cependant, forcer des activations ayant des moyennes nulles et des écart-types unitaires peut limiter le potentiel des réseaux de neurones. Cet effet est facilement éliminé en permettant au réseau de choisir lui-même les moyennes et écart-types des activations. La *batch normalisation*

remplace donc les activations d'une *mini-batch* $\boldsymbol{H}$ par les activations $\gamma \boldsymbol{H}' + \beta$, où β et γ sont la moyenne et l'écart-type choisis par le réseaux. Ils font partie des paramètres appris.

Ainsi, plutôt que de dépendre des interactions complexes de la couche, la moyenne et l'écart-type de la sortie de la couche sont directement paramétrisées, et donc facilement ajustées par le réseau. En effet, la moyenne et l'écart-type ne dépendent plus d'une multitude de paramètres, ils dépendent uniquement des paramètres β et γ.

De plus, les statistiques de chaque *mini-batch* sont une estimation bruitée des statistiques réelles de l'ensemble des données d'entrainement. La normalisation par ces moyennes et écarts-types bruitées a un effet régularisateur (Goodfellow *et al.*, 2016).

En résumé, l'utilisation de la *batch normalisation* à la sortie d'une couche facilite l'entraînement des couches suivantes, car la moyenne et l'écart-type des activations de la couche sont facilement ajustés lors de l'apprentissage. L'utilisation de la *batch normalisation* permet donc d'utiliser un taux d'apprentissage beaucoup plus élevé, accélérant l'apprentissage.

2.2.6 Généralisation

À la section 2.2.5, nous avons discuté d'optimisation. L'optimisation consiste à ajuster les paramètres du modèle de sorte qu'il obtienne de bonnes performances sur les données d'entraînement, c'est-à-dire qu'il obtienne une faible erreur d'entraînement.

Lorsqu'il sera déployé, le modèle sera cependant appelé à traiter des données qu'il n'aura jamais observées auparavant. On appelle "généralisation" l'habileté du modèle à performer sur des données qui n'ont pas été observées lors de l'entraînement. De même, l'erreur de généralisation, ou de test, est l'erreur obtenue par le modèle sur ces données jamais observées auparavant. On mesure généralement l'erreur de généralisation à l'aide d'un ensemble de données mis à part, les données de test.

Le défi de l'algorithme d'entraînement consiste donc à observer certaines données pour entraîner le modèle, de sorte qu'il performe bien sur toutes les données, qu'elles aient été observées auparavant ou non.

2.2.6.1 Sous-apprentissage, surapprentissage et capacité

Ainsi, l'algorithme minimise l'erreur d'entraînement de façon directe. Lorsque l'algorithme ne parvient pas à réduire cette erreur suffisamment, on dit que le modèle est en sous-apprentissage.

Aussi, on espère que la minimisation de l'erreur d'entraînement fasse indirectement diminuer l'erreur de généralisation. Lorsque la différence entre l'erreur d'entraînement et l'erreur de généralisation est élevée, on dit que le modèle est en surapprentissage.

Les phénomènes de sous-apprentissage et de surapprentissage sont fortement liés au concept de capacité d'un modèle.

La capacité d'un modèle est son aptitude à représenter différentes fonctions. Elle est principalement limitée par l'espace d'hypothèse du modèle, c'est-à-dire l'ensemble de fonctions qu'il peut utiliser comme solution au problème.

Un modèle de faible capacité aura de la difficulté à modéliser un phénomène complexe. Cette situation provoque le sous-apprentissage. À l'inverse, un modèle ayant une capacité trop élevée ne devrait pas être utilisé pour modéliser un phénomène simple, afin d'éviter des problèmes de surapprentissage. En effet, le modèle pourrait utiliser sa capacité excédentaire pour tenter de modéliser le bruit dans les données d'apprentissage, causant ainsi des problèmes de généralisation.

Il est donc important de créer des modèles ayant une capacité adaptée au problème. Dans les réseaux de neurones, la capacité dépend de divers facteurs, comme le type d'opérations effectuées, le nombre de couches cachées ou le nombre de neurones sur les couches cachées.

2.2.6.2 Régularisation

Goodfellow *et al.* (2016) définissent la régularisation comme "n'importe quelle modification apportée à un algorithme d'apprentissage qui vise à réduire son erreur de généralisation sans affecter son erreur d'entraînement".

Le théorème *No Free Lunch* (Wolpert & Macready, 1997) stipule essentiellement qu'aucun algorithme d'apprentissage machine n'est universellement meilleur que tous les autres.

Les algorithmes d'apprentissages doivent donc être adaptés selon la tâche à effectuer (Goodfellow *et al.*, 2016). Ils sont adaptés à la tâche en forçant des préférences pour certaines solutions, qui décrivent mieux le phénomène à modéliser. Par exemple, la désintégration des poids (*weight decay*), ajoute au coût un terme proportionnel à la norme L^2 des poids, ce qui favorise les solutions dont les poids sont plus près de zéro.

De même, l'architecture d'un modèle détermine son espace d'hypothèse. Exclure des fonctions de l'espace d'hypothèse du modèle est équivalent à exprimer une préférence infiniment forte contre ces fonctions. L'utilisation de la convolution, par exemple, force l'apprentissage de fonctions locales, invariantes aux translations, ce qui est très adapté au traitment d'images.

Ces techniques cherchent toutes à guider l'algorithme d'apprentissage vers une solution représentant mieux le phénomène, afin d'obtenir une meilleure généralisation. Elles ont donc un effet régularisateur.

Mentionnons que l'opération *batch normalisation* (voir section 2.2.5.3), qui vise tout d'abord à améliorer l'optimisation, a un léger effet régularisateur.

2.2.6.3 Désintégration des poids

La régularisation par désintégration des poids ajoute au coût un terme proportionnel à la norme L^2 des poids. Cela pénalise les poids s'éloignant trop de zéro, limitant du même coup la capacité (Goodfellow *et al.*, 2016).

2.2.6.4 Arrêt précoce

La régularisation par arrêt précoce (*early stopping*) utilise les données de validation (voir section 2.2.7) pour mesurer l'évolution de la généralisation pendant l'entraînement. En effet, lors de l'entraînement, on observe généralement que l'erreur de validation atteint un minimum, puis commence à augmenter, tandis que l'erreur d'entraînement continue de diminuer. Le modèle entre donc en surapprentissage. La technique de l'arrêt précoce reconnaît cette situation, arrête l'entraînement et retourne les paramètres du modèle ayant obtenu l'erreur de validation minimale. L'arrêt précoce limite le nombre d'epochs, en se basant sur l'erreur de validation, ce qui limite la taille des ajustements qui pourront être apportés aux paramètres initiaux. Cette technique a donc un effet similaire à la régularisation L^2 (Goodfellow *et al.*, 2016).

2.2.6.5 Dropout

Le *dropout* est une technique de régularisation (voir section 2.2.6.2) versatile, peu coûteuse en calcul et en mémoire. Elle fut introduite par Srivastava *et al.* (2014). Elle consiste à forcer à zéro la valeur d'une proportion prédéfinie des neurones de chaque couche lors de l'entraînement. À chaque *epoch*, pour chaque observation, les neurones forcés à zéro sont sélectionnées aléatoirement, habituellement en multipliant par un masque binaire aléatoire. La probabilité que chaque neurone soit forcé à zéro est un hyperparamètre. En général, cette probabilité est fixée à 20% pour les valeurs en entrée et à 50% pour les neurones des couches cachées.

Forcer la valeur d'un neurone à zéro est équivalent à retirer le neurone du réseau. Cela signifie que pour chaque masque binaire, un différent modèle est entraîné. En première approximation, le *dropout* peut donc être vu comme une méthode de régularisation très similaire au *bagging* (Goodfellow *et al.*, 2016).

D'un autre côté, le *dropout* force le réseau à apprendre des caractéristiques robustes, valides peu importe quels neurones ont été forcés à zéro.

Goodfellow *et al.* (2016) donnent l'exemple d'un réseau de neurones qui détecte les visages. Lors de l'entraînement, un neurone h_i apprend à s'activer lorsqu'un nez est présent dans l'image. Lorsque ce neurone est forcé à zéro par le *dropout*, un autre neurone est forcé à apprendre, soit à détecter les nez de façon redondante, soit à détecter une autre caractéristique permettant d'identifier les visages, comme les bouches. Le réseau, qui se fiait uniquement sur le neurone qui détectait les nez, apprend à se fier également à l'autre neurone qui détecte les bouches. Le réseau apprend donc plus précisément ce qui constitue un visage. En effet, forcer à zéro un neurone détectant les nez est équivalent à effacer l'information qu'il y a un nez dans l'image. Il serait très difficile d'effacer cette information en ajoutant du bruit dans l'image directement. Le *dropout* se sert donc de la compréhension des données acquise jusqu'ici dans l'entraînement pour effectuer une destruction intelligente de l'information contenue dans une observation. Le réseau est ensuite forcé à apprendre à effectuer sa tâche même lorsque cette information n'est pas présente.

Pour finir, le *dropout* applique le bruit de façon multiplicative, et non additive. Si le bruit était appliqué de façon additive sur les neurones, le réseau apprendrait à augmenter la valeur des poids afin de réduire l'effet relatif du bruit. L'application multiplicative du bruit ne permet pas au réseau de contourner le problème de robustesse au bruit aussi facilement (Goodfellow *et al.*, 2016).

2.2.6.6 Quantité de données

En plus des techniques de régularisation, une autre méthode pour combattre le surapprentissage et améliorer la généralisation du modèle est d'augmenter le nombre de données d'entraînement. Plus l'algorithme d'entraînement a accès à une grande quantité de données, plus il est en mesure d'approximer correctement la fonction réelle. La collecte et l'annotation de données est cependant une tâche fastidieuse et coûteuse.

Dans le cadre de ce projet, nous n'avons pas utilisé de régularisation L^2. Nous nous sommes plutôt tournés vers l'arrêt précoce. Nous avons également utilisé le *dropout*. Aussi, nous avons

utilisé des réseaux convolutionnels et des réseaux récurrents, espérant que l'espace d'hypothèse de ces modèles corresponde mieux à la nature du problème à résoudre. De plus, nous avons utilisé l'opération *batch normalisation*, qui peut indirectement avoir un léger effet régularisateur.

2.2.7 Hyperparamètres et validation

Les hyperparamètres sont des variables qui permettent de contrôler le fonctionnement des algorithmes d'apprentissage. Certains d'entre eux, comme le nombre de couches par exemple, permettent d'ajuster la capacité du modèle. D'autres, comme le taux d'apprentissage par exemple, permettent de contrôler l'apprentissage.

Contrairement aux paramètres, les hyperparamètres ne devraient pas être optimisés sur les données d'entraînement. En effet, prenons l'exemple des hyperparamètres contrôlant la capacité du modèle. Si ces hyperparamètres étaient optimisés sur les données d'entraînement, la capacité sera maximisée, ce qui mènerait au surapprentissage (Goodfellow *et al.*, 2016).

La sélection des hyperparamètres requiert donc la mise à part d'un autre ensemble de données, les données de validation.

2.2.7.1 Validation par la méthode *holdout*

Nous avons donc trois ensembles de données : Les données d'entraînement, de validation et de test.

A. Les données d'entraînement servent à optimiser les paramètres ;
B. Les données de validation servent à optimiser les hyperparamètres. Elles peuvent donc servir à mesurer la généralisation des paramètres, mais pas des hyperparamètres ;
C. Les données de test n'ont servi à la sélection d'aucun paramètre ou hyperparamètre. Elles permettent donc de mesurer l'erreur de généralisation des paramètres et hyperparamètres sélectionnés.

La méthode *holdout* consiste tout simplement à séparer les données disponibles en ensembles d'entraînement, de validation et de test.

Avec cette méthode, seule une fraction des données sert à mesurer la généralisation. Or, un nombre d'observations plus petit augmente l'incertitude statistique. Lorsque peu de données sont disponibles, il vaut mieux utiliser des techniques permettant d'utiliser toutes les données pour mesurer la généralisation, comme la validation croisée à *k folds*.

2.2.7.2 Validation croisée à k *folds*

La validation croisée simple à *k folds* sépare les données en *k folds* mutuellement exclusifs. L'un des *folds* sert d'ensemble de données de validation, tandis que les $k-1$ *folds* restant servent d'ensemble de données d'entraînement. Cette opération est répétée *k* fois, en utilisant à chaque fois un *fold* différent comme ensemble de données de validation. La performance globale est la moyenne des *k* mesures de performances. Cette méthode permet donc d'utiliser plus efficacement les données disponibles, mais multiplie la quantité d'entraînements à effectuer par *k*. En général, l'utilisation de $k = 10$ folds est recommandée (Kohavi, 1995).

La partition des données en *k folds* peut être effectuée de plusieurs manières. Certains séparent les observations de façon totalement aléatoire. Certains s'assurent que les classes sont représentées de façon égale dans chaque *fold*. Pour certains problèmes, il faut s'assurer que toutes les observations produites par une même distribution soient regroupées dans le même *fold* (comme c'est le cas dans ce projet, voir 4.4).

La validation croisée simple à *k folds* sépare donc les données en ensembles d'entraînement et de validation de *k* façons différentes. Elle ne prévoit cependant pas de données de test pour mesurer la généralisation des hyperparamètres.

2.2.7.3 Validation croisée imbriquée

La validation croisée imbriquée permet de résoudre cette difficulté, en appliquant la validation croisée de façon récursive. Une première validation croisée simple à *k folds* est employée pour séparer les données en ensembles d'entraînement et de test. Une seconde validation croisée simple à *n folds* est ensuite appliquée sur l'ensemble d'entraînement, afin de le subdiviser en ensembles de d'entraînement et de validation. Ainsi, la validation croisée imbriquée est constituée de deux boucles imbriquées. Cette méthode permet de mesurer la généralisation des hyperparamètres de façon fiable, mais multiplie la quantité d'entraînements à effectuer par $n \cdot k$.

Dans le cadre de ce projet, nous utilisons la validation croisée à *k folds*.

2.2.8 Optimisation des hyperparamètres

L'optimisation des hyperparamètres est simplement réalisée en entraînant le modèle avec différentes combinaison de valeurs d'hyperparamètres et choisissant la combinaison qui a obtenu la meilleure performance sur les données de validation.

2.2.8.1 Optimisation par recherche en grille

Avec la recherche en grille, on choisit d'avance une liste de valeurs candidates pour chacun des hyperparamètres. On teste ensuite toutes les combinaisons possibles des valeurs candidates des hyperparamètres. On sélectionne ensuite les hyperparamètres ayant obtenu la meilleure erreur de généralisation.

2.2.8.2 Optimisation par recherche aléatoire

Avec la recherche aléatoire, on attribue une distribution de valeurs admissibles à chaque hyperparamètre. On effectue ensuite plusieurs entraînements en échantillonnant aléatoirement les valeurs des hyperparamètres dans ces distributions. On sélectionne ensuite les hyperparamètres ayant obtenu la meilleure erreur de généralisation.

L'étude de Bergstra & Bengio (2012) indique que cette technique est beaucoup plus efficace que la recherche en grille. Nous utilisons donc la recherche aléatoire dans le cadre de ce projet.

CHAPITRE 3

EXPÉRIENCES

Dans le cadre de ce mémoire, nous entraînons des réseaux de neurones à reconnaître les pathologies chez les nourrissons par analyse de leurs cris.

Trois types de réseaux de neurones sont comparés, pour la reconnaissance de plusieurs pathologies différentes. Nous testons également deux méthodes de partition différentes.

Le système de dépistage est entraîné de façon supervisée. Le schéma bloc de son fonctionnement est présenté à la figure 3.1.

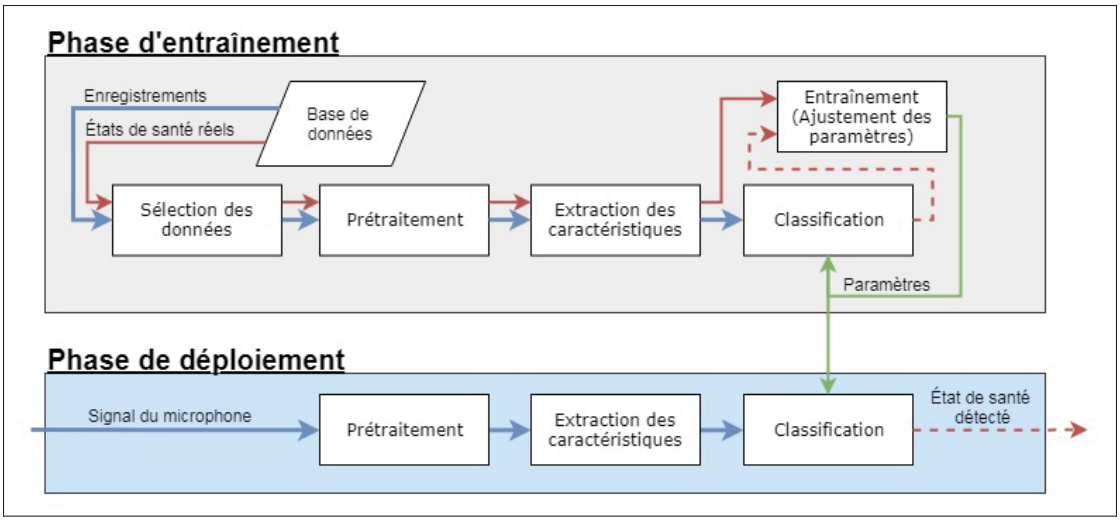

Figure 3.1 Schéma bloc global du système

Lors de la phase d'entraînement, des signaux de cris annotés, provenant d'une base de données, sont utilisés pour entraîner le modèle à associer les caractéristiques des cris à certaines pathologies.

Lors de la phase de déploiement, l'algorithme est en mesure d'évaluer l'état de santé d'un enfant en analysant son signal de cri.

3.1 Base de données

Nous utilisons les enregistrements de cris provenant de la base de données de l'École de Technologie Supérieure, qui a été récoltée avec le support financier de la fondation de Bill & Melinda Gates, dans le cadre du programme *Grand Challenges Explorations*.

Les échantillons constituant cette base de données ont été récoltés dans plusieurs hôpitaux au Canada et au Liban. Elle contient des enregistrements de cris d'enfants atteints de pathologies et d'enfants en santé. Elle contient des enregistrements provenant d'enfants nés à terme comme d'enfants prématurés.

Les cris ont été enregistrés par le personnel médical de l'hôpital avec un appareil portatif de marque *Olympus* à deux canaux, ayant une résolution de 16 bits et une fréquence d'échantillonnage de 44.1 kHz. Le microphone était placé à une distance variant entre 10 cm et 30 cm de la bouche de l'enfant. Les enregistrements ont été récoltés dans un environnement hospitalier. Ils contiennent donc du bruit de fond, des sons de machines, l'écho de conversation distantes, de portes qui ferment et ainsi de suite. La base de données représente donc bien l'environnement ou le système sera déployé (Abou-abbas *et al.*, 2015a).

Le personnel médical avait également la responsabilité de récolter plusieurs informations sur chaque enregistrement. Citons notamment :

A. La date et l'heure de l'enregistrement,
B. La date de naissance de l'enfant enregistré,
C. Les pathologies affectant l'enfant,
D. La cause des cris de l'enfant,
E. Le sexe de l'enfant,
F. L'ethnie de l'enfant,
G. L'âge gestationnel de l'enfant.

3.2 Sélection des données

3.2.1 Contraintes sur les données utilisées

Plusieurs études indiquent qu'à la naissance, le cri n'est pas contrôlé de façon consciente, il est instinctif. Cependant, au fur et à mesure que l'enfant mature, son cri devient plus volitif et commence à refléter divers états psychophysiologiques, comme la faim, la détresse, la douleur etc. (Golub & Corwin, 1985), (Wolf, 1969), (Bell & Salter Ainsworth, 1972). La plupart des études s'intéressent donc aux cris néonataux, c'est à dire d'enfants âgés de moins d'un mois, car ils sont plus instinctifs et moins variables. Dans le cadre de cette étude, nous n'avons donc utilisé que les signaux provenant d'enfants âgés de 30 jours ou moins. De plus, nous n'avons utilisé que des cris provenant d'enfants à terme, car l'âge gestationnel de l'enfant a également un effet sur les cris (Michelsson, 1971), (Thodén *et al.*, 1985).

Le tableau 3.1 résume la quantité de données respectant ces contraintes dans la base de données.

Tableau 3.1 Quantité de données disponibles

Données disponibles	
Nombre de nourrissons	480
Nombre d'enregistrements	1301
Nombre d'expirations	68334
Durée totale	39:51:29
Durée totale des expirations	14:27:45

3.2.2 Sélection des pathologies étudiées

Une grande quantité de pathologies sont représentées dans la base de données. Nous avons décidé de travailler sur les pathologies listées par Wasz-Höckert *et al.* (1985) car leurs effets sur le spectrogramme des cris sont bien connus. Parmi ces pathologies, nous retrouvons l'asphyxie, le syndrome de Down, l'hyperbilirubinémie, l'hypoglycémie et la méningite dans notre base

de données. Le nombre d'échantillons disponibles pour la plupart d'entre elles est cependant très limité.

Nous avons donc décidé d'ajouter la détresse respiratoire à la liste des pathologies étudiées. Elle a probalement un effet moins évident sur les cris, mais elle est la pathologie la mieux représentée dans la base de données.

La quantité de données disponibles pour chacune des pathologies étudiées, ainsi que pour les enfants en santé, est résumée dans le tableau 3.2. Les expirations de moins de 400 ms ont été exclues du décompte.

Tableau 3.2 Données disponibles pour les pathologies étudiées

Pathologies	Nourrissons	Enregistrements	Expirations	Blocs (éch. de 400 ms)
Asphyxie	1	3	196	370
Syndrome de Down	1	3	92	173
Hyperbilirubinémie	7	21	893	1879
Hypoglycémie	3	10	409	906
Méningite	2	6	331	499
Détresse respiratoire	35	106	5621	9409
En bonne santé	305	800	43505	75284

De plus, nous souhaitons comparer les performances obtenues avec nos modèles à celles de Farsaie Alaie *et al.* (2016). Nous entraînerons donc nos modèles à reconnaître les mêmes pathologies.

Le tableau 3.3 résume les pathologies reconnues par le modèle de Farsaie Alaie *et al.* (2016) Les pathologies sont regroupées selon le système affecté. Les pathologies contenues dans le groupe *Other* ne sont pas spécifiées. Nous avons choisi d'inclure les pathologies de notre base de données qui intuitivement pourraient avoir un effet sur le cri (cette sélection n'est confirmée par aucune étude).

Tableau 3.3 Données disponibles pour les pathologies du système de référence

Groupe	Pathologies	Nourrissons	Enregistrements	Expirations	Blocs (éch. de 400 ms)
En santé	Aucune	305	800	43505	75284
Problèmes cardiaques	Tétralogie de Fallot	1	2	60	72
	Thrombose	1	4	155	252
	"Complex Cardio"	1	3	73	187
	Maladies congénitales	1	3	102	317
Désordres neurologiques	Sepsie	16	50	2514	4269
	Méningite	2	6	331	499
Maladies respiratoires	Détresse respiratoire	35	106	5621	9409
	Asphyxie	1	3	196	370
Maladies du sang	Hyperbilirubinémie	7	21	893	1879
	Hypoglycémie	3	10	409	906
Autres	Ankyloglossie	1	3	138	395
	Syndrome de Down	1	3	92	173
	Atrésie Duodénale	1	3	153	246
	Gastrochisis	1	3	210	341
	Hypothermie	1	3	161	405
	Jaunisse	9	25	1420	2412
	Insuffisance rénale	1	3	184	363
	Crise épileptique	1	3	72	247

3.2.3 Définition des *datasets* et de leurs classes

Des *datasets* sont ensuite créés à partir de la liste des pathologies étudiées.

Nous nous limitons à des tâches de classification binaire. Chaque *dataset* est constitué de deux classes. La première contient des cris d'enfants en santé. La seconde contient des cris d'enfants atteints de la pathologie ou du groupe de pathologies qu'on souhaite reconnaître.

Nous souhaitons comparer la diffculté de reconnaître chacune des pathologies étudiées. Nous créons donc un *dataset* différent pour chacune d'entre elles.

Nous créons aussi un *dataset* de dépistage, afin d'évaluer la difficulté de reconnaître qu'un enfant est affecté par au moins une pathologie parmi les pathologies listées par Wasz-Höckert *et al.* (1985).

Nous tentons également de reproduire le *dataset* de Farsaie Alaie *et al.* (2016), afin de pourvoir comparer nos modèles aux leurs. Nous créons donc un *dataset* dont les classes sont définies de façon similaire.

Les définitions des *datasets* et de leurs classes sont résumées par la table 3.4.

Tableau 3.4 Définition des *datasets*

Nom du dataset	Définition	
	Classe 1	Classe 2
Asphyxie	En santé	Asphyxie
Syndrome de Down	En santé	Syndrome de Down
Hyperbilirubinémie	En santé	Hyperbilirubinémie
Hypoglycémie	En santé	Hypoglycémie
Méningite	En santé	Méningite
Dépistage	En santé	Asphyxie Syndrome de Down Hyperbilirubinémie Hypoglycémie Méningite
Détresse respiratoire	En santé	Détresse respiratoire
Référence	En santé	Tétralogie de Fallot Thrombose "Complex cardio" Maladies cardiaques congénitales Sepsie Méningite Détresse respiratoire Asphyxie Hyperbilirubinémie Hypoglycémie Ankyloglossie Syndrome de Down Atrésie duodénale Gastrochisis Hypothermie Jaunisse Insuffisance rénale Crise épileptique

3.3 Prétraitement

Les étapes du prétraitement sont présentées à la figure 3.2

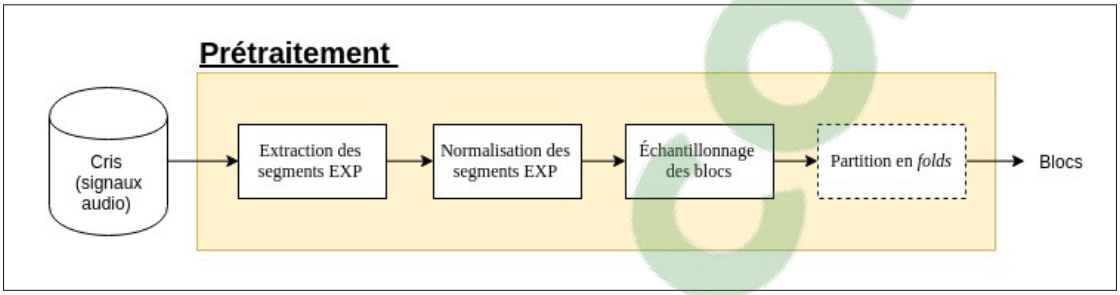

Figure 3.2 Schéma bloc du prétraitement

3.3.1 Segmentation

Abou-abbas *et al.* (2015a), Abou-abbas *et al.* (2015b), Abou-abbas *et al.* (2015c) et Abou-abbas *et al.* (2016) ont étudié la tâche de segmentation automatique des cris de nourrissons. Ils ont utilisé la base de données de l'École de Technologie Supérieure pour entraîner leurs algorithmes à reconnaître automatiquement différents types de segments dans les cris de nourrissons.

Dans le cadre de ce projet, nous utilisons uniquement les segments expiratoires voisés, car ils sont les plus communs et les plus importants pour la discrimination des pathologies.

Puisque leur algorithme était entraîné de façon supervisée, (Abou-abbas *et al.*, 2016) n'ont eu d'autre choix que de segmenter manuellement tous les signaux de la base de données. Ils ont effectué la segmentation à l'aide du logiciel *Wavesurfer* (Sjölander & Beskow, 2000).

Lors de la phase d'entraînement, nous utilisons cette segmentation manuelle pour récupérer les segments expiratoires voisés. Lors de la phase de déploiement, un algorithme comme celui développé par (Abou-abbas *et al.*, 2016) pourrait être utilisé pour extraire les segments expiratoires voisés.

3.3.2 Normalisation des segments

Les segments EXP (expiratoires voisés) sont normalisés, de sorte que le signal formé par la concaténation des segments EXP d'un même fichier ait une moyenne nulle et un écart-type unitaire. Nous avons appliqué cette normalisation de sorte que les expirations de tous les fichier aient, en moyenne, la même énergie.

3.3.3 Échantillonnage des blocs

Les segments sont ensuite découpés en blocs (ou échantillons) de durée fixe. Ce sont ces blocs que le système apprendra à classifier.

Cette approche est inspirée de l'architecture classique de traitement de cris de nourrissons, introduite par Petroni *et al.* (1995) pour la reconnaissance de la cause des cris, puis adaptée par Orozco-García & Reyes-García (2003b) pour la reconnaissance de pathologies.

La durée des échantillons est un paramètre important. Utiliser une durée trop élevée diminuerait considérablement le nombre d'échantillons disponibles pour l'entraînement. En contrepartie, réduire la durée des échantillons diminue la quantité d'informations qu'ils contiennent. On cherche donc un compromis entre la qualité et la quantité des échantillons disponibles.

La majorité des chercheurs échantillonnent l'enregistrement entier. Orozco-García & Reyes-García (2003b) par exemple découpent les enregistrements en échantillons de trois secondes. De leur côté, Petroni *et al.* (1995), Orozco-García & Reyes-García (2003a), Reyes-Galaviz & Reyes-García (2004), Suaste-Rivas *et al.* (2004a), Barajas-Montiel & Reyes-García (2006), Santiago-Sánchez *et al.* (2009), Rosales-Pérez *et al.* (2011), Zabidi *et al.* (2010), Sahak *et al.* (2012) et plusieurs autres découpent les enregistrements en échantillons d'une seconde.

Leurs échantillons ne sont cependants pas de très bonne qualité, car ils proviennent de l'enregistrement entier, qui contient plusieurs segments inutiles à la reconnaissance de pathologies, comme les silences.

De leur côté, Reyes-Galaviz *et al.* (2008) segmentent manuellement les enregistrements en *cry units*, puis les découpent en échantillons de 400 ms. Bien que ces échantillons soient plus courts, on s'attend à ce qu'ils soient de meilleure qualité, car seuls les segments pertinent des enregistrements sont échantillonnés.

Nous employons une approche similaire à celle de Reyes-Galaviz *et al.* (2008), en n'échantillonnant que les segments expiratoires voisés.

Sachant que la plupart des expirations voisées disponibles dans la base de donnée sont très courtes, comme on peut l'observer sur l'histogramme présenté à la figure 3.3, nous choisissons d'utiliser des échantillons de 400 ms. Cette durée représente un bon compromis entre la quantité d'échantillons disponibles (seules environ 22% des expirations dans la base de données durent moins de 400 ms, comme le montre l'histrogramme cumulatif à la figure 3.4) et leur qualité (Reyes-Galaviz *et al.* (2008) ont également utilisé cette durée et ont obtenu de bon résultats).

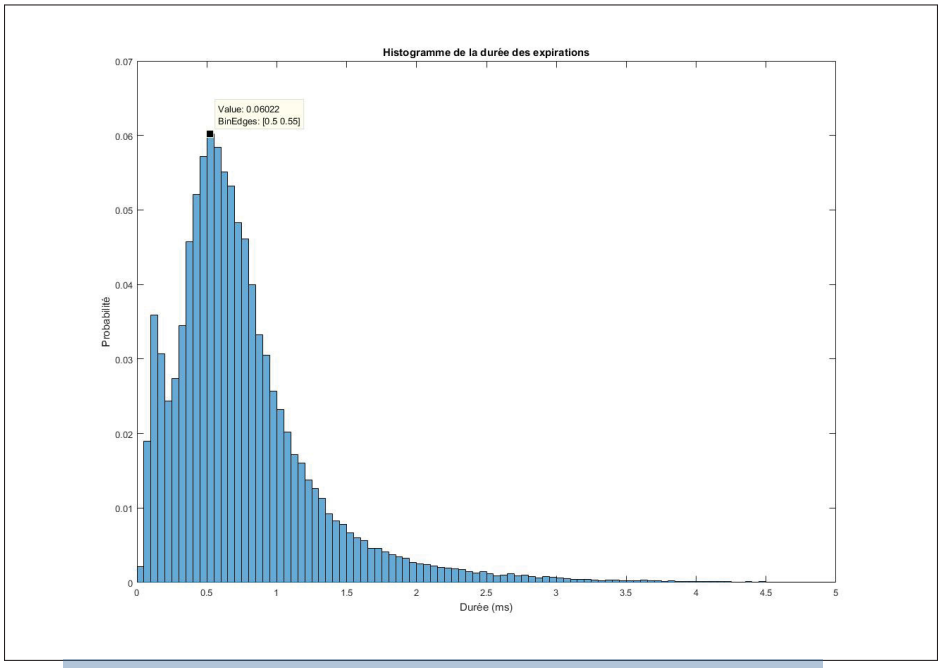

Figure 3.3 Histogramme de la durée de toutes les expirations voisées disponibles dans la base de données

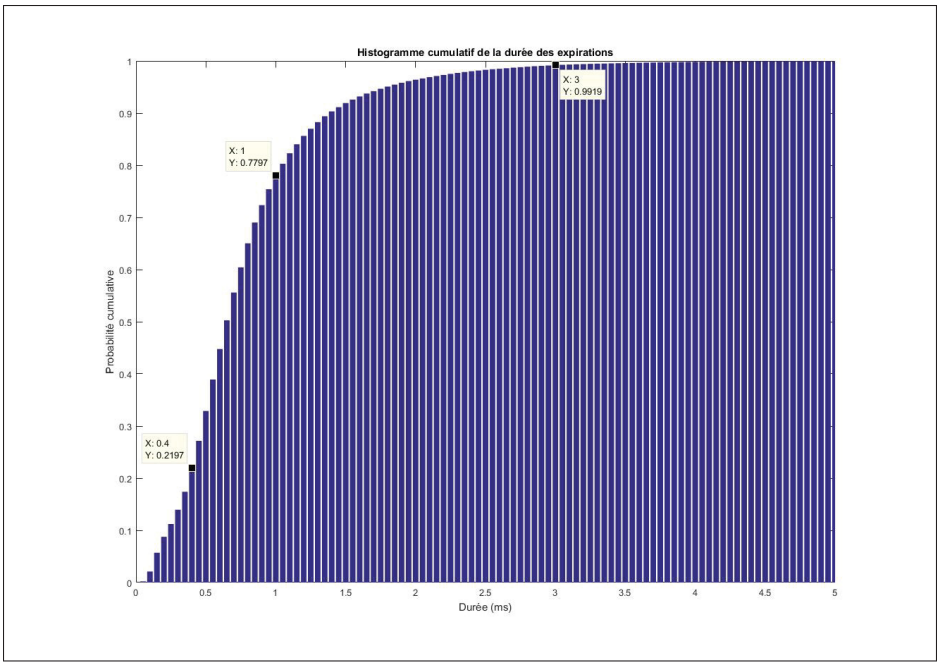

Figure 3.4 Histogramme cumulatif de la durée de toutes les expirations voisées disponibles dans la base de données

3.3.4 Partition en k *folds*

Nous utilisons la validation croisée à *k folds* pour optimiser les hyperparamètres et comparer les modèles. La partition en *folds* est uniquement nécessaire au processus d'entraînement. Lors de la phase de déploiement, aucune partition n'est nécessaire.

Nous comparons deux méthodes de partition des données en *k folds* :

A. Avec la première méthode, les échantillons sont répartis aléatoirement en *k folds* de taille égale, peu importe le fichier ou le nourrisson dont ils proviennent. Cette méthode pourrait introduire des biais lors de la validation. Ce détail important est discuté plus loin (section 4.4.1) ;

B. Avec la seconde méthode, les échantillons sont répartis aléatoirement dans les *folds*, avec la contrainte que deux *folds* différents ne puissent pas contenir d'échantillons provenant du même nourrisson. Pour s'assurer de respecter cette contrainte, l'algorithme assigne tous les échantillons d'un même nourrisson dans un seul *fold*.

Cette contrainte limite cependant le nombre de *folds* maximal. En effet, une classe contenant 5 nourrissons, par exemple, peut être divisée en un maximum de 5 *folds*. De même, un *dataset* constitué de deux classes, l'une contenant 5 nourrissons et l'autre 3, ne peut être divisé qu'en un maximum de 3 *folds*.

Nous utilisons toujours le nombre de *folds* maximal possible, qui correspond au nombre de nourrissons disponibles dans la classe contenant le moins de nourrissons. Par contre, lorsque ce nombre dépasse k (c'est-à-dire lorsque toutes les classes contiennent plus de k nourrissons), nous utilisons k *folds*.

Autrement dit, pour un *dataset* de n classes, où q_i correspond au nombre de nourrissons dans la classe i, alors nous séparons les données en $\min(k, \min(q_1, \ldots, q_n))$ *folds*.

Cette méthode permet d'éviter les biais de validation. Les *folds* n'auront cependant pas tous la même taille.

Dans tous les cas, l'algorithme de partition s'assure que les classes sont balancées dans chaque *fold*.

La figure 3.5 montre un exemple de partition selon les deux méthodes, pour $k = 4$. Il y a 3 nourrissons dans la première classe et 5 dans la seconde. Le nombre d'échantillons pour chaque nourrisson varie.

Avec la première méthode, chaque classe de chacun des 4 *folds* contient 12 échantillons assignés aléatoirement. Les *folds* ont donc tous la même taille, et les classes sont balancées.

Avec la seconde méthode, seuls 3 *folds* sont générés, car la classe 1 ne contient que 3 nourrissons. Afin de s'assurer que deux *folds* ne partagent pas des échantillons provenant d'un même nourrisson, les nourrissons A, D et E sont assignés au premier *fold*, les nourrissons B,F et G au second et les nourrissons C, H, I au troisième. Le nombre d'échantillons dans chaque *fold* est maximisé tout en maintenant la balance entre les classes. Les *folds* n'ont cependant pas tous la même taille. Notez que dans le cas ou la classe pathologique contient plusieurs pathologies, cette méthode ne s'assure pas que les pathologies soient représentées dans les mêmes

proportions d'un *fold* à l'autre. Cette erreur de méthodologie est discutée lors de l'analyse des résultats, à la section 4.4.

La partition des données de chacun des *datasets* utilisés dans le cadre de ce projet, pour les deux méthodes de partition, est résumée dans deux tables, situées en annexe II.

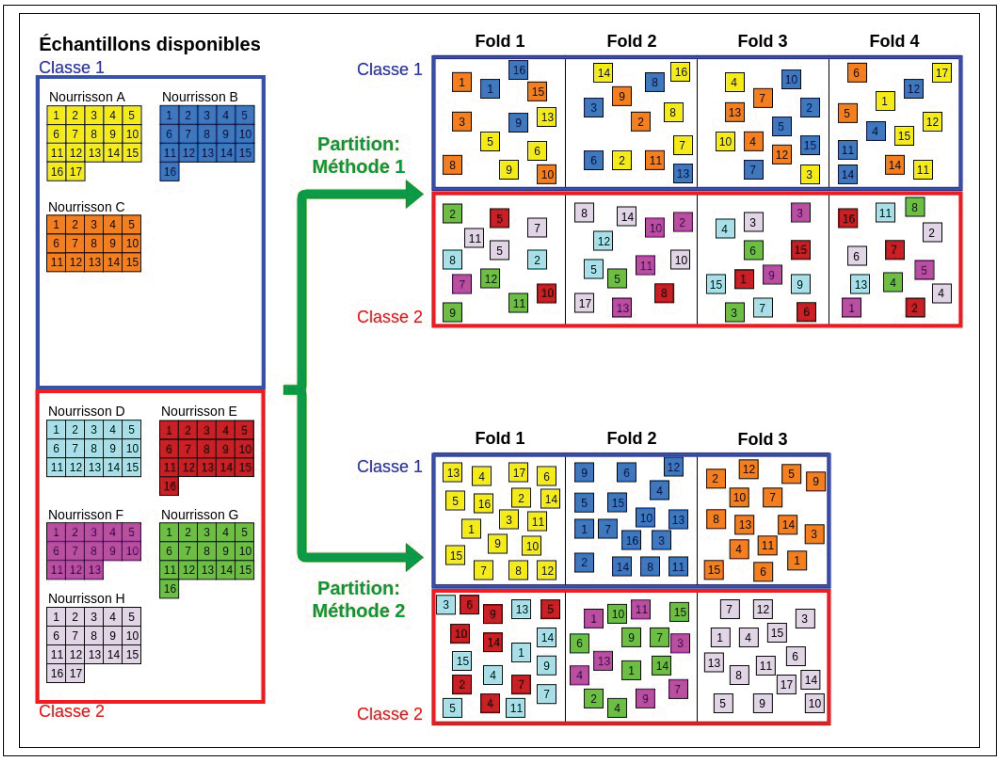

Figure 3.5 Comparaison des deux méthodes de partition à l'aide d'un exemple.

3.4 Extraction des attributs

Les diverses étapes de l'extraction des MFCC, qui serviront d'attributs, est représentée à la figure 3.6

Dans le cadre de ce projet, nous utilisons l'extraction des attributs classique développée par le groupe de recherche mexicain (Orozco-García & Reyes-García, 2003b), mentionnée à la section 1.3.3.1.

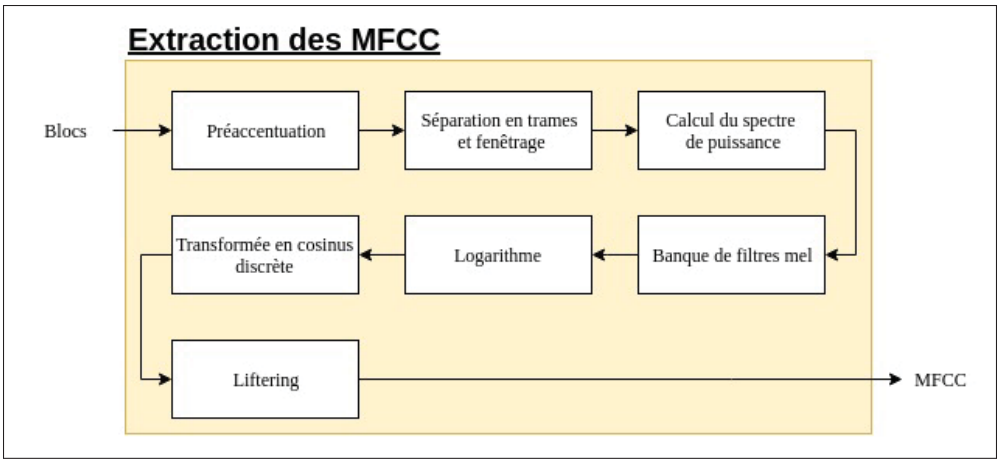

Figure 3.6 Schéma bloc de l'extraction des MFCC

Les chercheurs utilisent généralement le logiciel *Praat* (Boersma & van Heuven, 2001) pour extraire les MFCC. Nous avons utilisé la librairie de Ellis (2005), écrite en Matlab (The MathWorks Inc., 2016), pour extraire les MFCC. Cette librairie imite le calcul des MFCC par le logiciel HTK (Young *et al.*, 2013).

Les MFCC de chaque échantillon sont calculés selon la procédure suivante :

A. Le signal est d'abord filtré par un filtre de préaccentuation. Nous avons utilisé le filtre $H(z) = 1 - 0.97z^{-1}$;

B. Le signal est ensuite séparé en trames, puis chaque trame est multipliée par une fenêtre. Nous avons utilisé des trames d'une durée de 50 ms, sans recouvrement, avec la fenêtre de *hanning* ;

C. Le spectre de puissance de chaque trame est calculé. On obtient ainsi le spectrogramme de puissance ;

D. Une banque de 20 filtres mel, répartis sur une plage allant de 0 Hz à 4000 Hz, est utilisée pour calculer le spectrogramme de puissance mel ;

E. Le logarithme naturel du spectrogramme de puissance mel est calculé ;

F. Les 16 premiers coefficients de la transformée en cosinus discrète de chaque trame du cepstrogramme de puissance mel sont calculés ;

G. Un *lifter* est appliqué sur les coefficients. Nous utilisons une fenêtre sinusoïdale d'ordre $L = 22$;

H. Pour finir, le premier coefficient de chaque trame est remplacé par l'énergie totale de la trame.

La table 3.5 résume les hyperparamètres de la sélection des données, du prétraitement et de l'extraction des attributs. Les hyperparamètres surlignés en bleu sont variés manuellement. Les hyperparamètres non-surlignés ne sont pas variés.

Tableau 3.5 Hyperparamètres de la sélection des données, du prétraitement et de l'extraction des attributs

Étape	Hyperparamètre	Valeur
Sélection des données	Âge des nourrissons lors de l'enregistrement	1-30 jours
	Âge gestationnel des nourrissons	$\geq$ 37 semaines et 2 jours (né à terme)
	Définition des classes	Une des X définitions, voir REF
Prétraitement	Normalisation des expirations	Oui
	Durée des blocs (échantillons)	400 ms
	Incrément des blocs (échantillons)	400 ms
	Méthode de partition des données en *k folds*	Méthodes 1 ou 2, voir REF
Extraction des caractéristiques	Préaccentuation	Oui, filtre $H(z) = 1 - 0.97z^{-1}$
	Durée des trames	50 ms
	Incrément des trames	50 ms
	Fenêtre	*hamming*
	Type de spectrogramme	Puissance
	Plage de fréquence	0-4000 Hz
	Nombre de filtres mel	20
	Nombre de coefficients	16
	Ordre du lifter	22
	Remplacer le premier coefficient par l'énergie	Oui

3.5 Classification

Nous avons testé trois différent types de modèles : Des MLP, des réseaux convolutionnels (CNN) et des réseaux récurrents (LSTM).

En raison de la très grande quantité de calculs requis pour entraîner les réseaux, nous avons uniquement testé des réseaux à une seule couche cachée.

Dans les prochaines sections, les tables 3.6, 3.7 et 3.8 résument les hyperparamètres des trois modèles testés. La table 3.9 dresse la liste des hyperparamètres de l'entraînement. Dans toutes ces tables, les hyperparamètres surlignés en vert sont optimisés par recherche aléatoire. Les hyperparamètres non-surlignés n'ont pas été variés ou optimisés. La notation $[a,b]$ signifie un échantillonnage sur une distribution uniforme, allant de a à b. La notation $c^{[a,b]}$ réfère à un échantillonnage uniforme sur une échelle exponentielle de base c, allant de c^a à c^b.

Pour tous les modèles, le nombre de neurones (ou de *kernels*) par couche cachée a été varié entre 8 et 256. À 8 neurones, la capacité du réseau est déjà très limitée. Utiliser 256 neurones ou *kernels* commence à être coûteux en temps d'entraînement.

Pour ce qui du *dropout*, Srivastava *et al.* (2014) recommandent un taux de 0.2 en entrée et de 0.5 pour les autres couches. Nous avons donc testé les plages $[0-0.2]$ et $[0-0.5]$.

3.5.1 Modèle MLP

Un exemple, respectant l'architecture MLP employée dans le cadre de ce projet, est représenté à la figure 3.7.

Chaque couche cachée est constituée d'abord d'un *dropout*, puis d'une tranformation affine entièrement connectée, suivie d'une activation *ReLU*, puis d'une opération de *batch normalisation*.

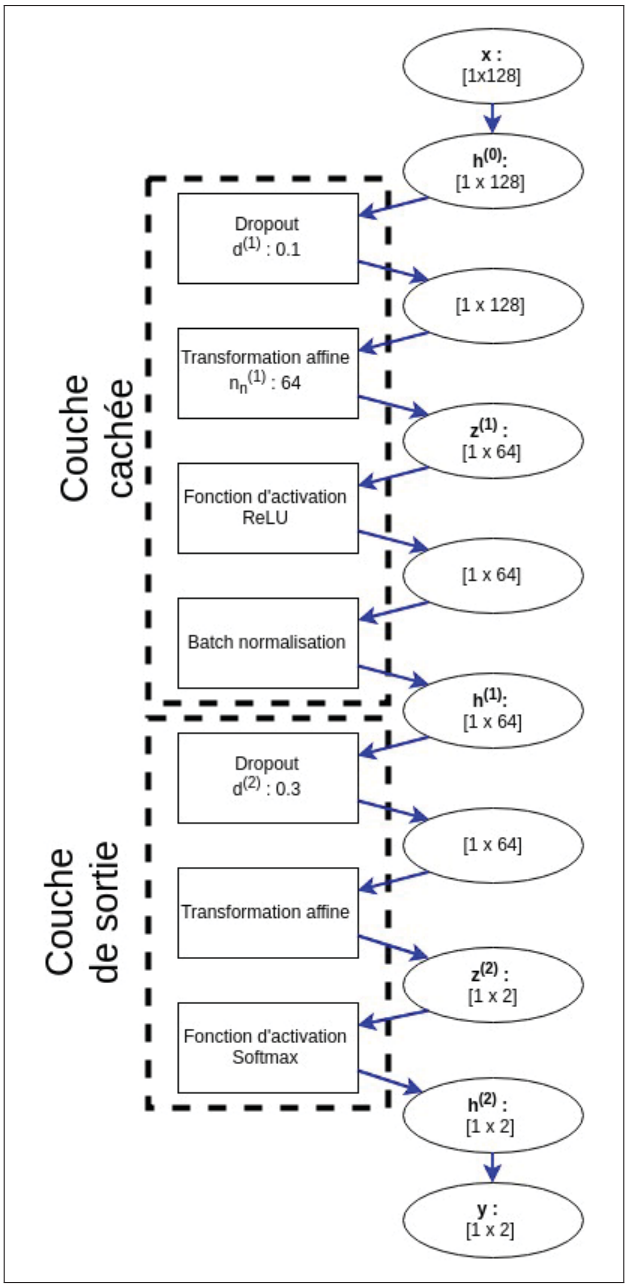

Figure 3.7 Exemple de modèle MLP respectant l'architecture employée dans le cadre de ce projet

La couche de sortie est constituée d'abord d'un *dropout*, suivi d'une transformation affine dont le nombre de neurones correspond au nombre de classes. Une activation *softmax* vient ensuite normaliser les probabilités ainsi calculées.

Mentionnons que dans le cas du modèle MLP, l'ACP a été appliquée sur les échantillons. [1]

Le tableau 3.6 résume les hyperparamètres du modèle MLP.

Tableau 3.6 Hyperparamètres du modèle MLP

Hyperparamètre	Symbole	Valeur
Nombre de couches cachées	L	1
Nombre de neurones de chaque couche cachée	$n_n^{(\ell)} \quad 1 \leq \ell \leq L$	$2^{[3,8]}$
Taux de dropout en entrée	$d^{(1)}$	$[0, 0.2]$
Taux de dropout des autres couches	$d^{(\ell)} \quad 2 \leq \ell \leq L+1$	$[0, 0.5]$

Seuls le nombre de neurones de la couche cachée ainsi que les taux de *dropout* ont été optimisés par recherche aléatoire.

Dans l'exemple présenté à la figure 3.7, la couche cachée de 64 neurones a un taux de *dropout* de 0.1. La couche de sortie a quant à elle un taux de *dropout* de 0.3.

3.5.2 Modèle CNN

Un exemple, respectant l'architecture CNN utilisée dans le cadre de ce projet, est représenté à la figure 3.8.

Chaque couche cachée est constituée d'abord d'un *dropout*, puis d'une convolution 2D, suivie d'un *max pooling* 2D, puis d'une activation *ReLU*, puis d'une opération de *batch normalisation*. Nous n'utilisons aucun *padding* lors de la convolution.

La couche de sortie est constituée d'un *dropout* suivi d'une transformation affine dont le nombre de neurones correspond au nombre de classes. Une activation *softmax* vient ensuite normaliser les probabilités ainsi calculées.

1. Cette opération était utilisée par Orozco-García & Reyes-García (2003b) pour réduire le nombre d'attributs. De notre côté cependant, nous avons conservé tous les attributs. Cette opération n'est donc pas réellement nécessaire. Elle élimine les corrélations entre les MFCC, qui sont déjà approximativement décorrélés grâce à la transformée en cosinus discrète.

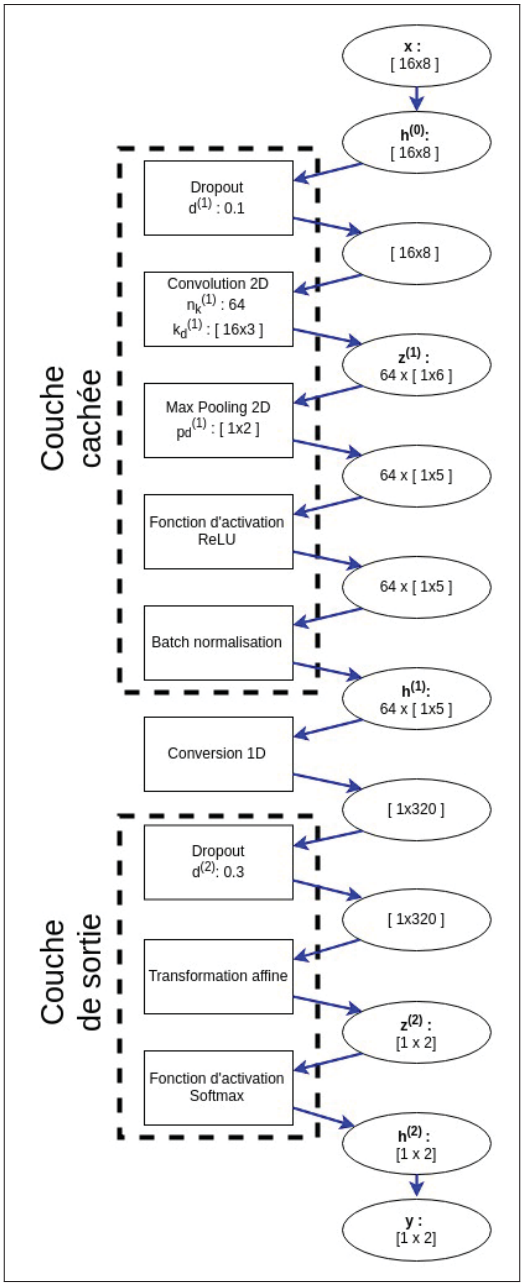

Figure 3.8 Exemple de modèle CNN respectant l'architecture employée dans le cadre de ce projet

Le tableau 3.7 résume les hyperparamètres du modèle CNN.

Seuls les taux de *dropout*, ainsi que le nombre de *kernels* et la dimension temporelle des *kernels* et du *max pooling* de la couche cachée, ont été optimisés par recherche aléatoire.

Tableau 3.7 Hyperparamètres du modèle CNN

Hyperparamètre	Symbole	Valeur
Nombre de couches cachées	L	1
Nombre de *kernels* de chaque couche cachée	$n_k^{(\ell)}$ $1 \leq \ell \leq L$	$2^{[3,8]}$
Dimension des *kernels* de chaque couche cachée	$k_d^{(\ell)}$ $1 \leq \ell \leq L$	16 x $[1,4]$
Incrément des *kernels* de chaque couche cachée	$k_i^{(\ell)}$ $1 \leq \ell \leq L$	1 x 1
Dimensions du *max pooling* de chaque couche cachée	$p_d^{(\ell)}$ $1 \leq \ell \leq L$	1 x $[1,4]$
Incréments du *max pooling* de chaque couche cachée	$p_i^{(\ell)}$ $1 \leq \ell \leq L$	1 x 1
Taux de dropout en entrée	$d^{(1)}$	$[0,0.2]$
Taux de dropout des autres couches	$d^{(\ell)}$ $2 \leq \ell \leq L+1$	$[0,0.5]$

Dans l'exemple présenté à la figure 3.8, la couche cachée de 64 *kernels* de dimension 16x3 est suivie d'un *max pooling* de 1x2. Elle a un taux de *dropout* de 0.1, alors qu'il est de 0.3 pour la couche de sortie.

En forçant la dimension cepstrale des *kernels* à être égale au nombre de MFCC, nous avons en réalité réalisé une convolution 1D dans le temps. En effet, il n'y a pas de gain à partager les paramètres entre les MFCC, car ils ne sont pas structurés en grille. La dimension temporelle, par contre, représente une séquence. Le partage des paramètres d'une trame à l'autre est donc justifié. Nous avons varié la taille de la dimension temporelle des *kernels* entre 1 et 4.

Le sous-échantillonnage permet de réduire la quantité de calculs à effectuer en éliminant une partie de l'information. Nous n'avons pas utilisé cette technique, car la quantité d'information est déjà limitée. Nous avons donc utilisé un incrément de 1x1 pour la convolution et le *max pooling*.

3.5.3 Modèle LSTM

Un exemple, respectant l'architecture récurrente LSTM utilisée dans le cadre de ce projet, est représenté à la figure 3.9.

Chacune des couches cachées, excepté la dernière, est constituée d'une cellule LSTM retournant des séquences. La dernière couche cachée est constituée d'une cellule LSTM retournant

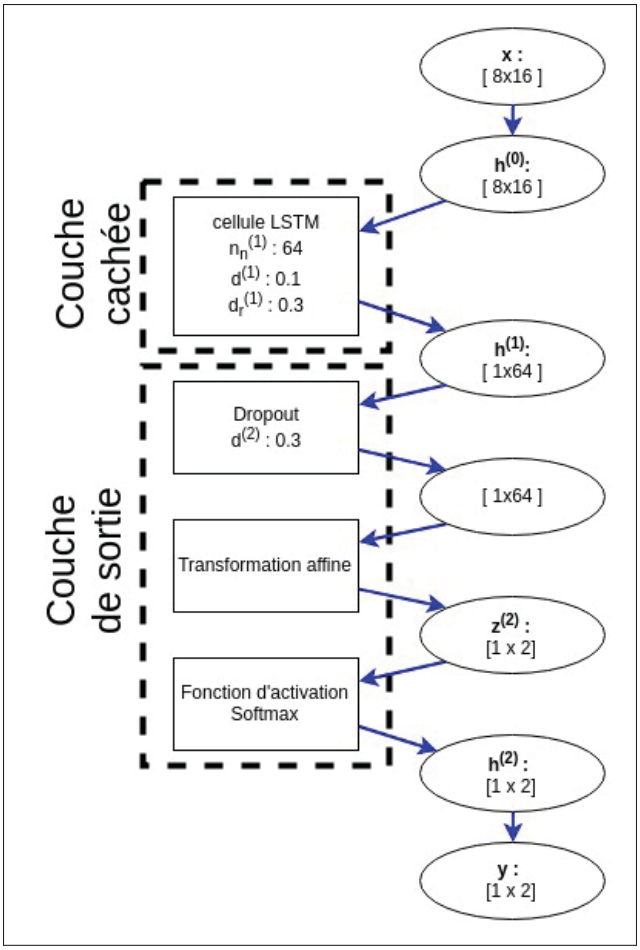

Figure 3.9 Exemple de modèle LSTM respectant l'architecture employée dans le cadre de ce projet

uniquement le dernier élément de sa séquence de sortie. Les cellules LSTM ont des activations récursives *sigmoïdes* et une activation *tanh* en sortie.

La couche de sortie est constituée d'un *dropout*, puis d'une transformation affine dont le nombre de neurones correspond au nombre de classes. Une activation *softmax* vient ensuite normaliser les probabilités ainsi calculées.

Le tableau 3.8 résume les hyperparamètres du modèle LSTM.

Tableau 3.8 Hyperparamètres du modèle LSTM

Hyperparamètre	Symbole	Valeur
Nombre de couches cachées	L	1
Nombre de neurones de chaque couche cachée	$n_n^{(\ell)} \quad 1 \leq \ell \leq L$	$2^{[3,8]}$
Fonction d'activation	f_a	*tanh*
Fonction d'activation récurrente	f_{ar}	*sigmoïde*
Taux de dropout en entrée	$d^{(1)}$	$[0, 0.2]$
Taux de dropout récurrent en entrée	$d_r^{(1)}$	$[0, 0.5]$
Taux de dropout des autres couches	$d^{(\ell)} \quad 2 \leq \ell \leq L+1$	$[0, 0.5]$
Taux de dropout récurrent des autres couches	$d_r^{(\ell)} \quad 2 \leq \ell \leq L+1$	$[0, 0.5]$

Seuls le nombre de neurones de la couche cachée, ainsi que les taux de *dropout* et les taux de *dropout* récurrents, ont été optimisés par recherche aléatoire.

L'exemple présenté à la figure 3.8 contient une couche cachée de 64 neurones ayant en entrée des taux de *dropout* et de *dropout* récurrent de 0.1 et 0.3, respectivement. Le taux de *dropout* de la couche de sortie est de 0.3.

Nous utilisons les fonctions d'activation les plus communes. Nous varions les taux de *dropout* récurrents entre 0 et 0.5.

3.6 Entraînement

3.6.1 Algorithme d'entraînement des classifieurs

Les réseaux sont tous entraînés par descente du gradient optimisée par l'algorithme Adam (Kingma & Ba, 2015). Les gradients sont calculés par rétropropagation. Nous utilisons l'entropie croisée comme fonction de coût.

Les hyperparamètres de l'entraînement sont présentés dans le tableau 3.9.

Tableau 3.9 Hyperparamètres de l'entraînement

Hyperparamètre	Valeur
Taux d'apprentissage initial	$10^{[-3.5,-4.5]}$
Taille des *batch*	32
Nombre d'epochs	100

3.6.2 Évaluation des classifieurs

La performance d'un *fold* est constituée de la précision de validation au coût de validation minimal. La performance globale du modèle est la moyenne des performances de chaque *fold*, pondérée par la taille de chaque *fold*.

3.7 Outils

Nous avons utilisé plusieurs programmes et librairies pour réaliser ce projet.

La sélection et le prétraitement des données et l'extraction des attributs ont été réalisés avec Matlab (The MathWorks Inc., 2016). Les calculs ont été effectués avec la librairie de Ellis (2005).

Les modèles ont étés construits et entraînés en langage Python (Python Software Foundation, 2018) avec la librairie Keras (Chollet & Others, 2015), qui est basée sur Tensorflow (Abadi *et al.*, 2015).

Pour ce qui est de l'entraînement des modèles, les calculs ont été effectués sur le supercalculateur *Hélios* de l'Université Laval, sous la gouverne de Calcul Québec et Calcul Canada. L'exploitation de ce supercalculateur est financée par la Fondation canadienne pour l'innovation (FCI), le ministère de l'économie, de la science et de l'innovation du Québec (MESI), et le Fonds de recherche du Québec - Nature et technologies (FRQ-NT).

CHAPITRE 4

PRÉSENTATION DES RÉSULTATS ET DISCUSSION

4.1 Entraînement et optimisation des hyperparamètres

Au cours de ce projet, 36600 entraînements ont été requis pour optimiser les hyperparamètres de chaque modèle, pour chaque *dataset*, par recherche aléatoire, pour deux méthodes de partition des données différentes.

4.1.1 Exemple de test

Chaque recherche aléatoire était constituée de 100 tests. À chaque test, les hyperparamètres étaient échantillonnés aléatoirement dans les plages de valeurs spécifiées dans les tables 3.6, 3.7, 3.8, 3.9 présentées précédemment.

Lors de chaque session d'entraînement, la progression du coût et de la précision sur les données d'entraînement ainsi que sur les données de validation était sauvegardée, pour chaque *fold*. Ces informations étaient ensuite utilisées pour calculer la métrique de performance.

Les figures 4.1 et 4.2 montrent un exemple de courbes d'entraînement produites par un test lors de l'optimisation par recherche aléatoire des hyperparamètres d'un MLP pour la tâche de reconnaissance de l'hyperbilirubinémie, en utilisant la première méthode de partition. Dans cet exemple de test, l'échantillonnage aléatoire a produit un taux d'apprentissage de 0.00029 et un réseau ayant 180 neurones sur la couche cachée, avec des taux de *dropout* de 0.07 en entrée et de 0.25 sur la couche de sortie.

On observe qu'au début, le coût de validation est plus faible que le coût d'entraînement. Cela est dû principalement au *dropout*, qui vient diminuer la performance lors de l'entraînement, mais n'affecte pas la validation.

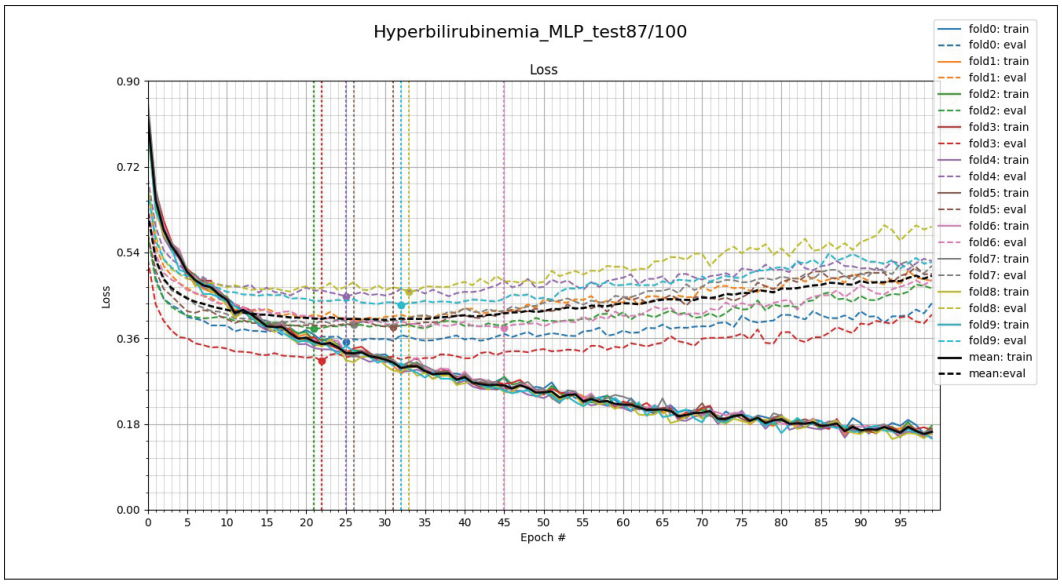

Figure 4.1 Courbes d'entraînement d'un MLP sur le *dataset* de l'hyperbilirubinémie, démontrant l'évolution du coût pour chaque *fold*. Les marqueurs et les lignes verticales indiquent la position des coûts de validation minimums de chaque *fold*

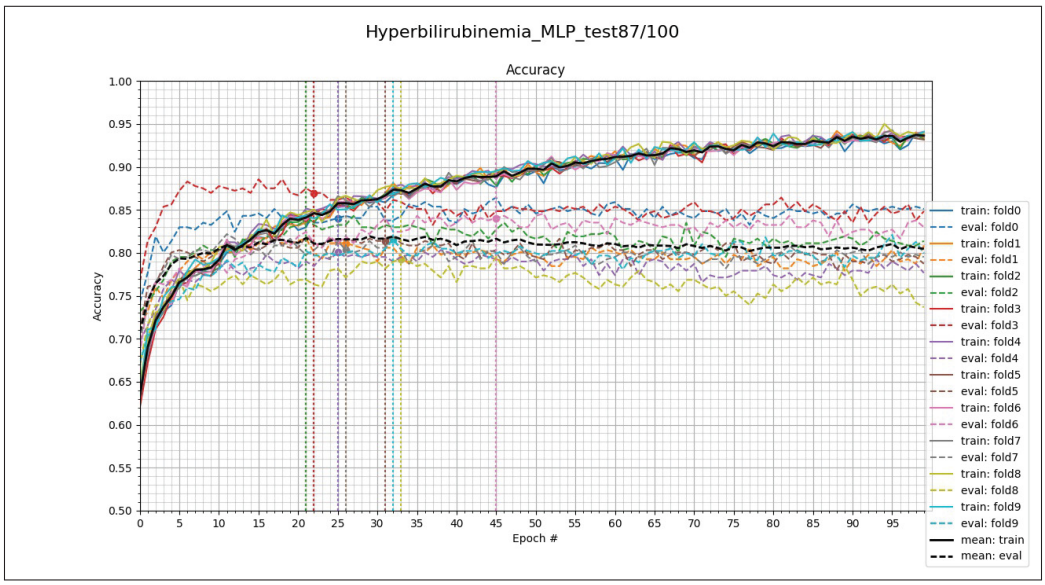

Figure 4.2 Courbes d'entraînement d'un MLP sur le *dataset* de l'hyperbilirubinémie, démontrant l'évolution de la précision de classification pour chaque *fold*. Les lignes verticales indiquent la position du coût de validation minimal de chaque *fold*

De plus, avec la librairie Keras, le coût d'entraînement rapporté à chaque *epoch* est la moyenne des coût des *mini-batchs*. Étant donné que les paramètres du modèle sont mis à jour à chaque *mini-batch*, cette moyenne surestime le coût d'entraînement réel (Chollet & Others, 2015).

Comme expliqué à la section 3.6.2, nous mesurons la précision de validation lorsque le coût de validation est au minimum, pour chaque *fold*. Nous effectuons la moyenne de ces précisions, pondérée par le nombre de données de validation de chaque *fold*. Ce test, par exemple, a obtenu une performance de 82.25%.

4.1.2 Exemple de recherche aléatoire

Le test montré en exemple à la section précédente a obtenu le meilleur résultat parmi 100 tests effectués. La figure 4.3 présente la distribution des performances obtenues lors de cette optimisation par recherche aléatoire. Dans cet exemple, on observe que la médiane des performances est d'environ 81.3%. Le meilleur test, présenté à la section précédente, a obtenu une performance de 82.25%.

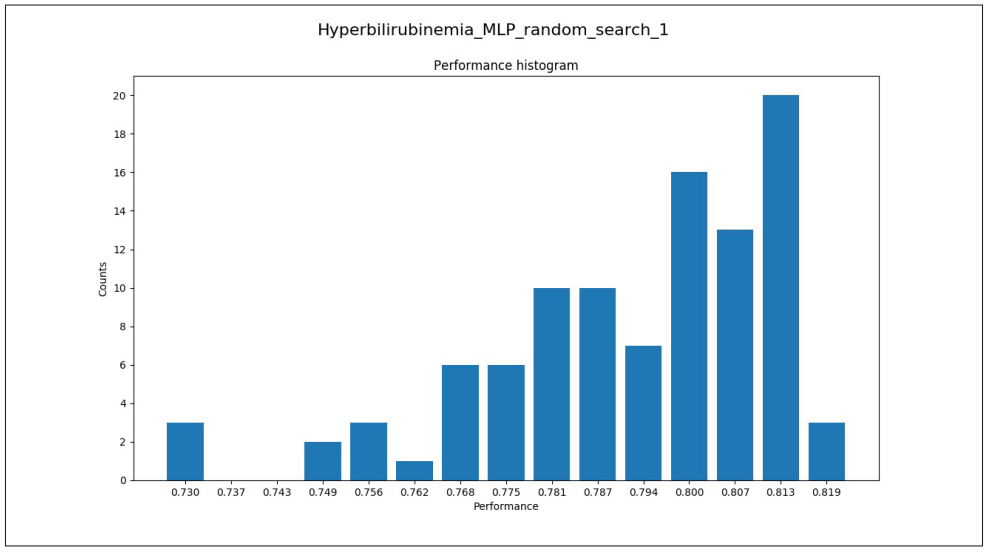

Figure 4.3 Distribution des performances obtenues lors des 100 tests effectués pour l'optimisation des hyperparamètres d'un MLP pour la tâche de reconnaissance de l'hyperbilirubinémie

4.2 Présentation des résultats

Les meilleures performances obtenues lors de l'optimisation des hyperparamètres de chaque modèle, sur chaque *dataset*, avec les deux méthodes de partition différentes, sont présentées dans le tableau 4.1.

Tableau 4.1 Meilleure performance obtenue pour chacun des modèles, sur chaque *dataset*, avec les deux méthodes de partition

Nom du dataset	Échantillons	Méthode de partition 1			Méthode de partition 2		
		MLP	CNN	LSTM	MLP	CNN	LSTM
Syndrome de Down	346	86,71%	92,20%	89,88%			
Asphyxie	740	95,27%	97,97%	96,76%			
Méningite	998	98,00%	98,90%	98,60%	94,69%	94,39%	88,68%
Hypoglycémie	1812	89,18%	93,10%	93,71%	60,04%	55,63%	59,16%
Hyperbilirubinémie	3758	82,25%	85,55%	86,62%	65,01%	67,38%	61,02%
Dépistage	7654	81,76%	85,67%	87,50%	64,27%	68,62%	62,26%
Détresse respiratoire	18818	75,28%	79,59%	81,36%	64,45%	66,46%	63,19%
Référence	45484	73,44%	76,69%	80,50%	65,87%	67,53%	64,59%

Ces résultats seront analysés dans les sections suivantes.

4.3 Quantité de données disponibles

La taille de chacun des *datasets* est comparée à la figure 4.4.

Les classes sont parfaitement balancées dans chacun des *datasets*. Autrement dit, chaque *dataset* contient autant d'échantillons sains que pathologiques. Les échantillons sains abondent (75284 échantillons provenant de 305 nourrissons sont disponibles dans la base de données). Ce sont donc les échantillons pathologiques qui limitent la taille des *datasets*.

La quantité de données disponibles dans chacun des *datasets* détermine le degré de confiance que nous avons envers les résultats. Le nombre d'échantillons est important, mais le nombre d'enfants dont il proviennent l'est tout autant, sinon plus.

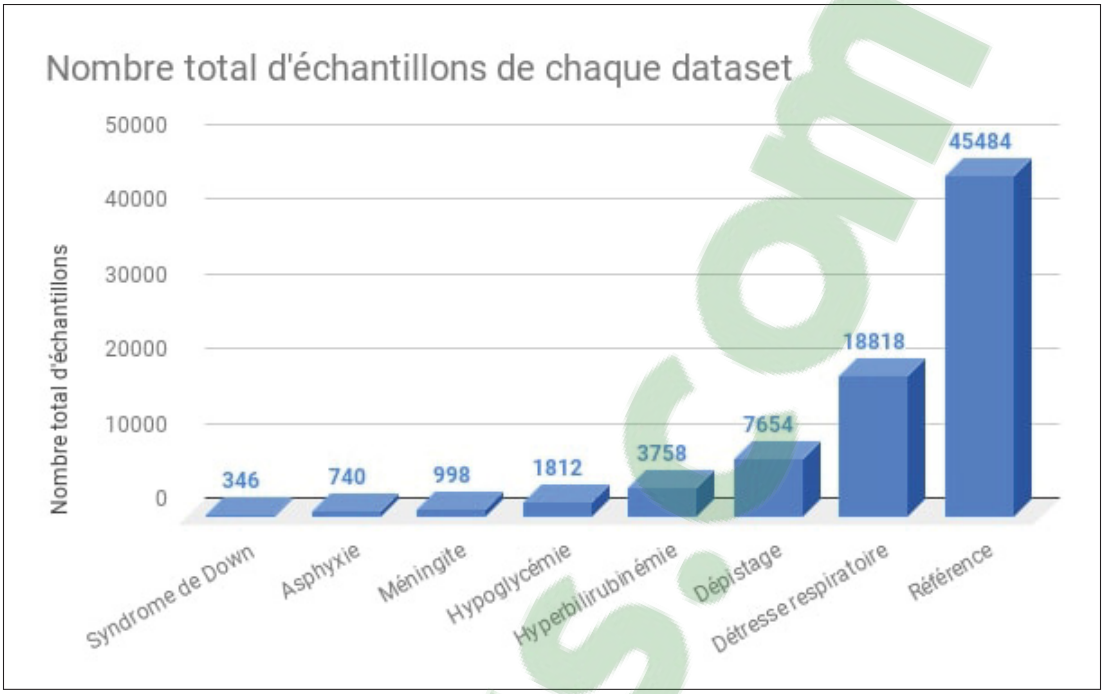

Figure 4.4 Nombre d'échantillons dans chacun des *datasets*

La rareté de certaines pathologies néonatales, comme le syndrome de Down, l'asphyxie, la méningite ou l'hypoglycémie limite la quantité d'enregistrements disponibles. Il est très difficile d'entraîner des réseaux de neurones à partir de si peu de données avec succès. Les résultats présentés dans cette étude pour ces 4 pathologies sont donc très incertains et ne permettent pas de tirer de conclusions.

Pour ce qui est de l'hyperbilirubinémie, nous estimons que 1879 échantillons pathologiques provenant de 7 nourrissons différents permettent un degré de confiance minimal envers les résultats obtenus.

De même, le *dataset* "dépistage", combinant les 5 pathologies les plus rares, contient 3827 échantillons pathologiques provenant de 14 nourrissons différents, pour un niveau de confiance modéré. Nous devons cependant tenir compte du fait que les pathologies ne sont pas représentées également dans ce *dataset*. En effet, 49% des échantillons pathologiques représentent l'hyperbilirubinémie, 24% l'hypoglycémie, 13% la méningite, 9% l'asphyxie et seulement 5% le syndrome de Down.

Le *dataset* "détresse respiratoire" contient 9409 échantillons pathologiques provenant de 35 nourrissons, une quantité suffisante pour produire des résultats crédibles.

Pour ce qui est du *dataset* "référence", il est le plus gros, avec 22742 échantillons pathologiques provenant de 84 nourrissons différents. Les résultats obtenus avec ce *dataset* sont très fiables. Il faut cependant prendre en compte que près de 41% des échantillons pathologiques de ce *dataset* représentent la détresse respiratoire.

Un autre effet important de la quantité de données est son effet régularisateur. En effet, augmenter la quantité de données d'entraînement réduit l'erreur de généralisation (Goodfellow *et al.*, 2016). Les *dataset* contenant plus de données sont donc avantagés.

Les systèmes de reconnaissance automatisée de pathologies par analyse de cris de nourrissons pourraient certainement bénéficier de la construction de plus grandes bases de données. La collecte d'enregistrements de cris de nourrissons est cependant une tâche ardue et coûteuse, requiérant un suivi médical de chaque enfant étudié. De plus, étant donné que les cris sont des signaux d'origine humaine, plusieurs enjeux éthiques viennent compliquer la collecte et le partage des données.

4.4 Comparaison des méthodes de partition

Dans ce projet, nous avons évalué deux méthodes de partition des données.

Les figures 4.5, 4.6, et 4.7 permettent de comparer les performances obtenues pour les différentes méthodes de partition.

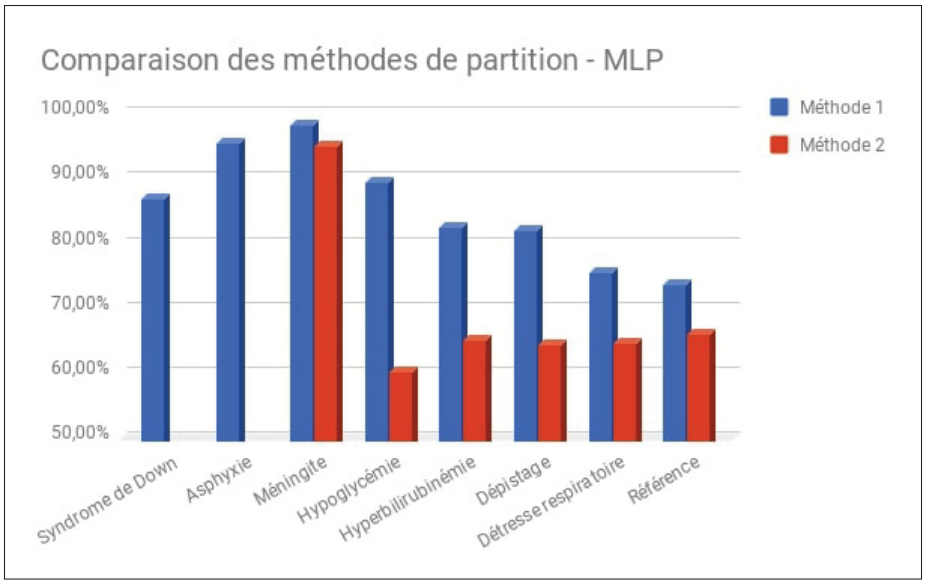

Figure 4.5 Comparaison des performances obtenues avec des modèles MLP pour deux méthodes de partitionnement

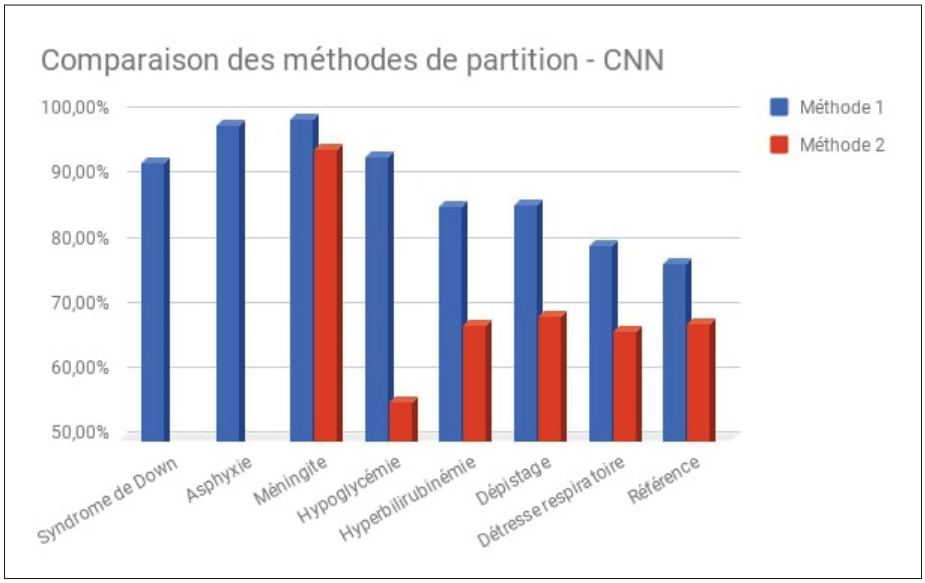

Figure 4.6 Comparaison des performances obtenues avec des modèles CNN pour deux méthodes de partitionnement

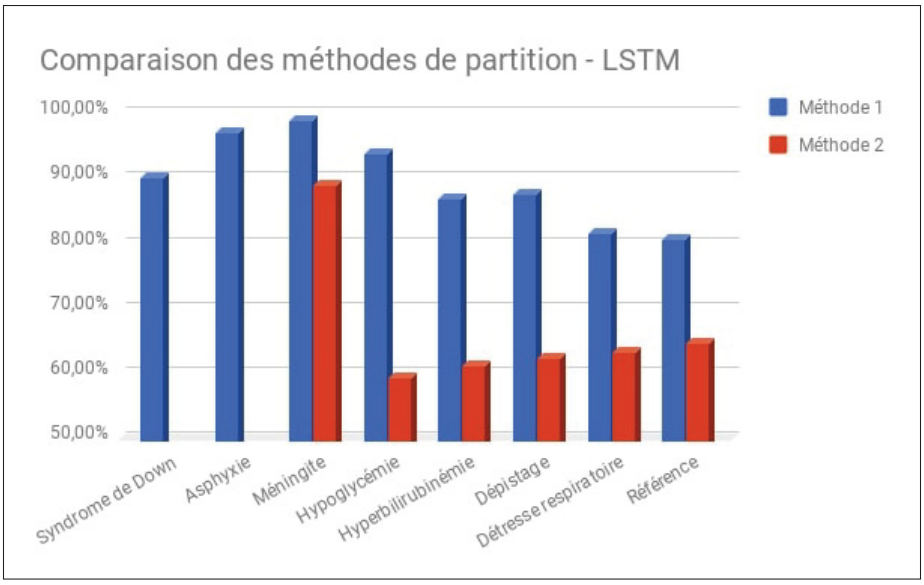

Figure 4.7 Comparaison des performances obtenues avec des modèles LSTM pour deux méthodes de partitionnement

On observe clairement que lorsque la seconde méthode de partition des données est employée, les performances diminuent radicalement, peu importe le modèle, pour tous les *datasets* excepté celui qui porte sur la reconnaissance de la méningite.

L'observation des courbes d'entraînement nous aide à comprendre ce phénomène. La figure 4.8 présente les courbes d'entraînement d'un MLP pour la reconnaissance de la détresse respiratoire avec la première méthode de partition. Les hyperparamètres ont été optimisés par recherche aléatoire.

La figure 4.9 présente les courbes d'entraînement du même MLP (c'est-à-dire ayant les mêmes hyperparamètres), pour la même tâche, mais entraîné en utilisant la seconde méthode de partition.

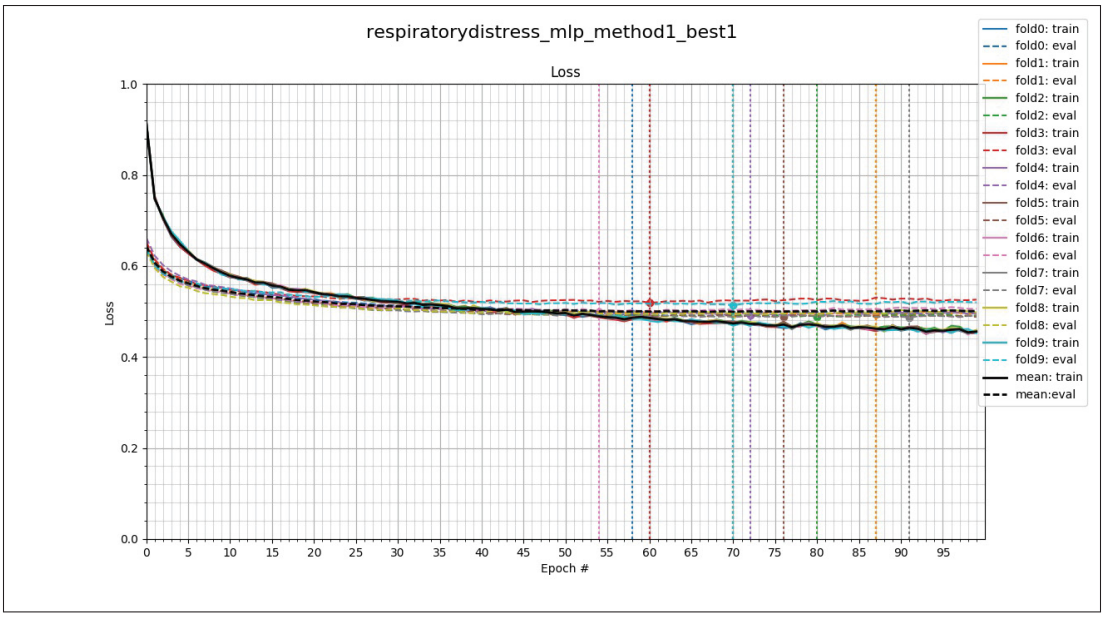

Figure 4.8 Courbes de coût de l'entraînement d'un MLP, avec la première méthode de partition

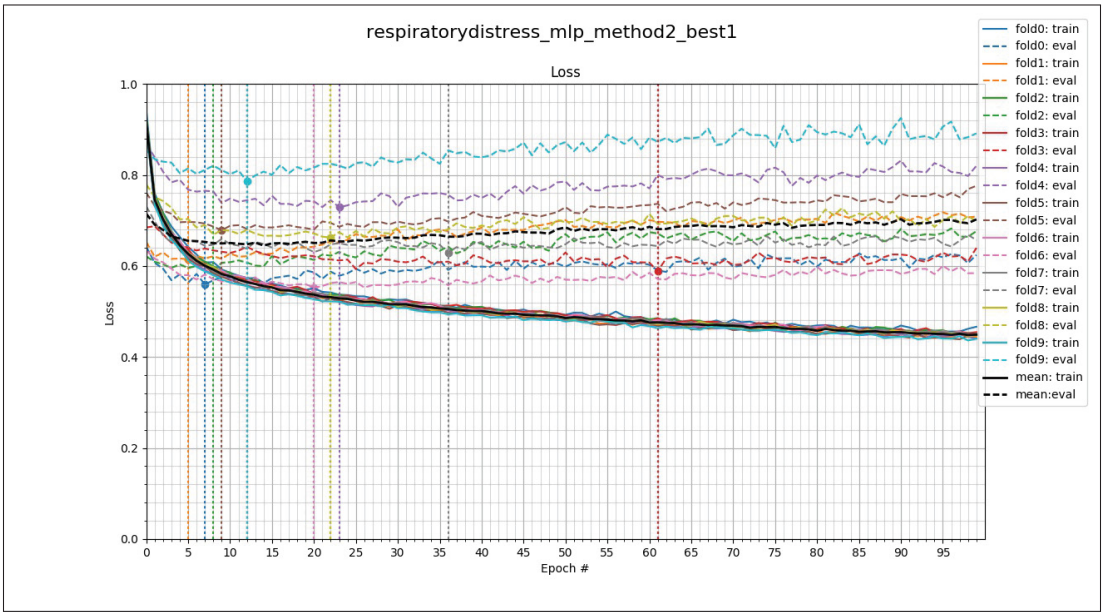

Figure 4.9 Courbe de coût de l'entraînement du même MLP, avec la seconde méthode de partition

A première vue, une comparaison entre les courbes d'entraînement pour les deux méthodes de partition semble indiquer que les modèles ont beaucoup plus de difficulté à généraliser

avec la seconde méthode, car le coût de validation minimum est atteint beaucoup plus tôt, et la valeur est beaucoup plus élevée. Nous expliquons néanmoins à la section 4.4.1 qu'en réalité, les performances obtenues avec la première méthode de partition sont biaisées et que tous nos modèles entrent rapidement en surapprentissage, peu importe la méthode de partition employée.

On note également une plus grande variabilité d'un *fold* à l'autre, pour les courbes d'entraînement comme pour les courbes de validation, avec la seconde méthode de partition. Lorsque la première méthode de partition est employée, les *folds* sont plus homogènes, chacun d'entre eux pouvant contenir des échantillons provenant de tous les nourrissons. Lorsque la seconde méthode est employée, les *folds* sont davantage hétérogènes, chacun caractérisé par le sous-ensemble unique de nourrissons dont proviennent ses échantillons. Ces grandes variations entre les *folds* viennent mettre en évidence l'individualité des cris de chaque nourrisson.

4.4.1 Biais de validation

Cette section vise à expliquer la dégradation de performance apparente lorsque la seconde méthode de partition est employée.

4.4.1.1 Surapprentissage

Lors de l'entraînement, l'algorithme d'apprentissage ajuste les paramètres du modèle de manière à minimiser le coût des données d'entraînement. Plusieurs "stratégies" pourraient être employées pour faire diminuer le coût d'entraînement. Il est difficile de prédire la laquelle sera employée.

Le modèle pourrait, par exemple, apprendre à reconnaître l'ambiance sonore dans les échantillons utilisés lors de l'entraînement et s'y fier pour effectuer ses décisions. Un tel modèle performerait bien sur les enregistrements observés lors de l'entraînement, mais généraliserait mal à d'autres enregistrements. On parle donc de surapprentissage des enregistrements. Nous

estimons qu'il s'agit d'une des stratégies les plus faciles à mettre en oeuvre, car différentes ambiances sonores produiront des attributs facilement distinguables.

De même, un modèle qui apprendrait à reconnaître les nourrissons représentés dans les données d'entraînement généraliserait mal à des nourrissons jamais observés lors de l'entraînement. On parle ici de surapprentissage des nourrissons. D'ailleurs, maintes expériences ont révélé que les parents sont en général capables de reconnaître le cri de leur propre enfant (Valanne *et al.*, 1967), (Morsbach & Bunting, 1979), (Gustafsson *et al.*, 2013). Cela signifie que l'identification de nourrissons par analyse de leurs cris est possible. Il est cependant difficile de prédire cette tâche est plus aisée que la reconnaissance des pathologies.

Nous souhaitons que le modèle apprenne à reconnaître les symptômes des pathologies, c'est-à-dire les motifs statistiquements plus communs ou rares dans les cris d'enfants atteints. Un tel modèle serait capable de reconnaître les pathologies, peu importe l'enregistrement ou le nourrisson dont proviennent les échantillons.

Nous croyons cependant que les modèles auront surtout tendance à surapprendre les enregistrements. Le surapprentissage des nourrissons pourrait aussi être non-négligeable.

4.4.1.2 Mesure biaisée de la généralisation

Avec la première méthode de partition, les échantillons sont séparés aléatoirement en ensembles d'entraînement et de validation. Il est pratiquement certain que parmi les échantillons d'un même enregistrement, certains servent à l'entraînement, alors que d'autres servent à la validation. Lorsque c'est le cas, on ne mesure pas correctement la généralisation du modèle à de nouveaux enregistrements. Autrement dit, un modèle souffrant de surapprentissage des enregistrements obtiendrait une bonne performance sur les données de validation, mais performerait très mal sur le terrain, où il serait confronté à des échantillons provenant de nouveaux enregistrements, jamais observés lors de l'entraînement.

De façon similaire, avec la première méthode de partition il est très probable que parmi les échantillons d'un même nourrisson, certains servent à l'entraînement, alors que d'autre servent à la validation, menant à une mesure incorrecte de la généralisation à de nouveaux nourrissons.

Les performances obtenues sur les données de validation avec la première méthode de partition surestiment donc la généralisation réelle du modèle à de nouveaux enregistrements et à de nouveaux nourrissons.

La seconde méthode de partition, quant à elle, s'assure que l'ensemble d'entraînement et de validation ne puissent contenir tous les deux des échantillons provenant d'un même nourrisson ou d'un même enregistrement. La performance sur les données de validation permet donc de mesurer la généralisation réelle du modèle.

Autrement dit, avec la première méthode de partition, on mesure essentiellement l'habilité du modèle à distinguer le groupe d'enregistrements (ou nourrissons) pathologiques du groupe d'enregistrements (ou nourrissons) sains dans les données d'entraînement, tandis qu'avec la seconde méthode, on mesure l'habileté du modèle à reconnaître les pathologies elles-mêmes.

4.4.2 Méthodes de partitions employées dans d'autres recherches

Il est rare que la méthode de partition des données en ensembles d'entraînement et de validation soit décrite précisément dans les articles de recherche. Il est donc difficile de juger des résultats. La formulation de certains auteurs porte cependant à croire qu'ils ont réparti leurs données selon une procédure similaire à ce que nous appelons "la première méthode de partition".

Orozco-García & Reyes-García (2003b) par exemple expliquent qu'ils ont alétoirement séparé leurs échantillons en 10 *folds* et mentionnent également que tous leurs *folds* contiennent exactement le même nombre d'échantillons.

De plus, plusieurs études emploient la base de données *Baby Chillanto* pour entraîner des modèles à reconnaître, soit la surdité, soit l'asphyxie, en se servant de la validation croisée à 10 *folds*.

Parmi les études produites par le groupe de recherche mexicain, c'est le cas de Orozco-García & Reyes-García (2003a), Orozco-García & Reyes-García (2003b), Barajas-Montiel & Reyes-García (2006), Santiago-Sánchez *et al.* (2009) et Rosales-Pérez *et al.* (2011). De même, dans le groupe de recherche malaisien, c'est le cas des études de Sahak *et al.* (2012), Hariharan *et al.* (2012b), Saraswathy *et al.* (2013), Wahid *et al.* (2016).

Le problème est que la base de données *Baby Chillanto* contient les cris de seulement 6 nourrissons atteints de surdité et de 6 nourrissons ayant souffert d'asphyxie.

On voit mal comment les échantillons provenant de 6 nourrissons peuvent être séparés en 10 *folds* sans que deux *folds* différents ne possèdent des échantillons provenant d'un même nourrisson.

Nous soupçonnons que ces études aient partitionné les échantillons sans tenir compte du nourrisson dont ils provenaient, ce qui vient remettre en question la validité des résultats rapportés. Ces soupçons viennent malheureusement aussi réduire notre confiance envers toutes les autres études produites par les groupes de recherche mexicain et malaisien.

Nous avons contacté M. Carlos Alberto Reyes-Garcìa, meneur du groupe de recherche mexicain, pour discuter de cette question. Nous sommes encore dans l'attente d'une réponse.

4.4.2.1 Partition pour le dépistage de plusieurs pathologies différentes

Lorsque la classe pathologique contient plusieurs pathologies, comme c'est le cas pour nos *datasets* "dépistage" et "référence", il aurait également fallu s'assurer que toutes les pathologies étudiées soient représentées dans les mêmes proportions, d'un *fold* à l'autre. Autrement dit, supposez un *dataset* dont la classe pathologique contient les pathologies A, B et C. Supposez que 70% des échantillons pathologiques dans la classe appartiennent à la pathologie A, 25% à la pathologie B et 5% à la pathologie C.

Idéalement, les données seraient partitionnées de manière à ce que ces proportions soient conservées dans chacun des *folds*, tout en s'assurant que deux *folds* ne puissent partager des échantillons provenant d'un même nourrisson.

Lorsque la première méthode de partition est employée, les proportions sont conservées, mais les nourrissons sont mélangés.

Lorsque la seconde méthode de partition est employée, les nourrissons sont correctement séparés, mais les proportions sont sévèrement inégales entre les *folds*. Il est même très probable qu'une pathologie ne soit pas représentée dans tous les *folds*.

Ainsi, la première méthode de partition est inadéquate, tandis que la seconde n'est adéquate que lorsqu'une seule pathologie est étudiée.

Par conséquent, tous nos *datasets* ont été partitionnés correctement avec la seconde méthode, excepté les *datasets* "dépistage" et "référence". Les résultats obtenus sur ces *datasets* sont donc peu représentatifs des performances réelles que notre modèle obtiendrait.

4.5 Comparaison des classifieurs

Nous avons utilisé la validation croisée simple à *k folds*, présentée à la section 2.2.7. Cette méthode mesure la généralisation des paramètres, mais pas celle des hyperparamètres, et surestime donc les performances réelles.

Nous avons effectué ce choix pour plusieurs raisons :

A. Nous cherchons à identifier les approches prometteuses et non à développer un produit fini. Les effet du surapprentissage des hyperparamètres devraient être négligeables par rapport aux comparaisons qu'on souhaite effectuer ;

B. Les recherches sur lesquelles nous nous sommes basés n'utilisaient pas d'ensemble de test non plus. Ils séparaient uniquement leurs données en ensembles d'entraînement et de validation ;

C. Mettre des données de côté aurait davantage réduit la quantité déjà limitée de données disponibles pour l'entraînement.

La performance des modèles est comparée aux figures 4.10 et 4.11

Les *datasets* sont présentés en ordre croissant du nombre d'échantillons.

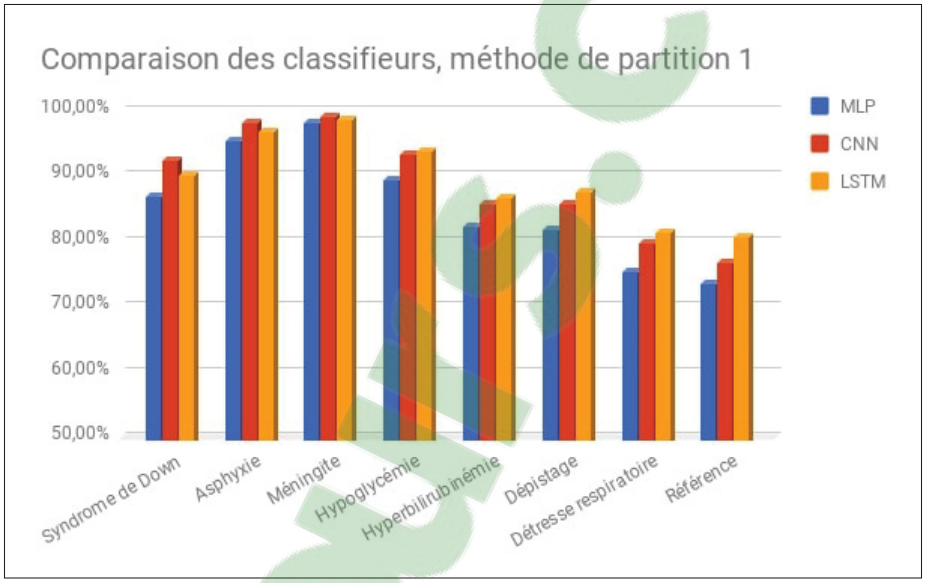

Figure 4.10 Comparaison de la performances des divers classifieurs sur différents *datasets*, avec la première méthode de partition

Avec la première méthode de partition, on observe que les LSTM obtiennent des résultats légèrement supérieurs pour les *datasets* contenant plus d'échantillons ("hypoglycémie", "hyperbilirubinémie", "détresse respiratoire", "dépistage" et "référence"), tandis que les CNN performent mieux pour les *datasets* ayant moins d'échantillons ("asphyxie", "syndrome de Down", "méningite"). Ces résultats sont cependant peu pertinents, car ils sont fortement biaisés et ne représentent pas la réelle généralisation des modèles. À partir de maintenant, nous n'analyserons que les résultats obtenus avec la seconde méthode de partition, qui n'est pas affectée par ce biais important.

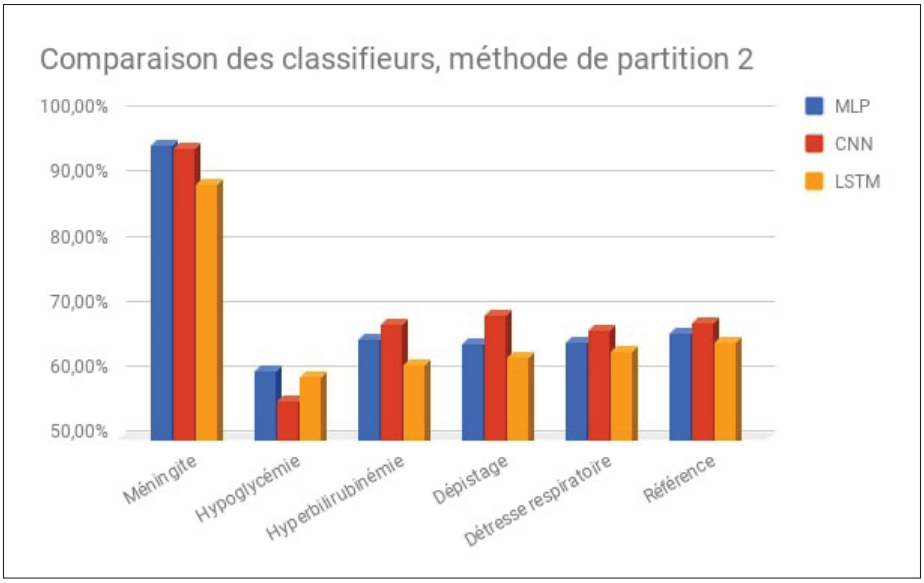

Figure 4.11 Comparaison de la performances des divers classifieurs sur différents *datasets*, avec la seconde méthode de partition

Avec la seconde méthode de partition, on constate que les CNN sont de peu supérieurs aux autres modèles pour les *datasets* contenants plus d'échantillons, alors que les LSTM obtiennent constamment les pires performances.

Les MLP surpassent les CNN sur les *datasets* contenant moins d'échantillons, c'est-à-dire les *datasets* "hypoglycémie" et "méningite". Cependant, comme il a été discuté à la section 4.3, la taille limitée de ces *datasets* ne permet pas de tirer de conclusion.

Il semble donc que les CNN obtiennent les meilleures performances. Ce type de modèle est bien adapté au problème, car il est spécialisé dans le traitement de signaux en grille, ce qui inclut les courtes séquences.

Il est normal que les MLP obtiennent des performances inférieures, car sont des réseaux génériques, ne tirant pas avantage de la structure des signaux.

Les LSTM, quant à eux, sont conçus pour le traitement de séquences. Comment se fait-il qu'ils obtiennent les pires performances ? En réalité, leur spécialité est le traitement de séquences

comportant des interactions à long terme. Or, les séquences traitées dans le cadre de ce projet sont très courtes. Il est possible que la complexité inutile de ce type de modèle ait été nuisible.

4.6 Comparaison des tâches

De plus, il est difficile de comparer la difficulté de reconnaître différentes pathologies, car le nombre d'échantillons disponibles varie grandement d'une pathologie à l'autre. Comme expliqué précédemment à la section 4.3, une augmentation de la quantité de données pourrait mener à une amélioration des performances. Nous espérons tout de même pouvoir identifier des tendances.

En observant le graphique 4.11, on remarque tout d'abord la précision élevée obtenue pour la reconnaissance de la méningite. En effet, contrairement aux autres *datasets*, la performance obtenue pour la méningite diminue très peu lorsqu'on passe de la première à la seconde méthode de partition. Il est possible que la méningite soit plus facilement reconnaissable que les autres pathologies. Il faut cependant se rappeler que les échantillons représentant la méningite ne proviennent que de 2 enfants, avec 3 enregistrements pour chacun d'entre eux. Les résultats pour cette pathologie ont donc peu de signification statistique. Il faudrait plus d'échantillons provenant de différents enfants pour pouvoir tirer une conclusion.

L'hypoglycémie semble être l'une des pathologies les plus difficiles à reconnaître. Cependant, le nombre réduit de données ne permet pas de se prononcer. En effet, les échantillons de l'hypoglycémie proviennent d'un total de 10 enregistrements de 3 nourrissons.

Malheureusement, nos résultats pour les *datasets* "dépistage" et "référence" sont peu fiables, étant donné que nous avons mal partitionné les données lorsque la classe pathologique contient plusieurs pathologies différentes.

Ainsi, seuls les résultats des *datasets* "hyperbilirubinémie" et "détresse respiratoire" peuvent réellement être comparés. Il est intéressant de constater que de meilleures performances ont été obtenues pour la reconnaissance de l'hyperbilirubinémie que pour celle de la détresse respira-

toire, et ce malgré une quantité de données d'entraînement largement inférieure (3758 contre 18818 échantillons). Cela nous porte à croire que l'hyperbilirubinémie est plus facile à reconnaître que la détresse respiratoire.

4.7 Comparaison avec le système de référence

Nous souhaitons comparer nos performances à celles obtenues dans l'étude de Farsaie Alaie *et al.* (2016). Comme expliqué à la section 1.3.3.3, ils se sont servi d'une version antérieure de notre base de données pour entraîner des GMM à reconnaître les pathologies chez les nourrissons. Afin de pouvoir comparer nos performances à celles qu'ils ont obtenues, nous avons créé un *dataset* ayant sensiblement la même définition de classes que le leur, le *dataset* "référence". Malheureusement, nous avons mal partitionné les données dans ce *dataset*. Nous présentons tout de même la comparaison de ces résultats avec ceux du système de référence, mais rappelons que cette comparaison est peu fiable.

Le système de Farsaie Alaie *et al.* (2016) est plus élaboré que le nôtre. Une première étape de dépistage détecte si l'enfant est en santé ou malade. Une seconde étape tente d'identifier le système affecté chez les enfants malades. La tâche effectuée à cette première étape est la même que la nôtre. Farsaie Alaie *et al.* (2016) utilisent cependant deux classifieurs, dont ils fusionnent les résultats pour prendre la décision finale. L'un des classifieurs analyse les segments expiratoires, l'autre les segments inspiratoires. Nous n'avons considéré que les segments expiratoires. Nous effectuons donc une comparaison avec le sous-système de Farsaie Alaie *et al.* (2016), c'est-à-dire leur classifieur de dépistage par analyse de segments expiratoires.

Leur classifieur est construit à base de GMM. Les paramètres des GMM ont étés calculés par adaptation des paramètres d'un GMM-UBM, modélisant les segments expiratoires de tous les nourrissons, qu'ils soient malades ou en santé. Quatre méthodes d'adaptation différentes ont été comparées. Le logarithme du rapport de vraisemblance (LLR) est utilisé pour prendre la décision finale. Deux versions différentes du LLR ont été testées, ainsi que deux durées de trames.

Farsaie Alaie *et al.* (2016) ont partitionné leurs données en 2 ensembles : l'ensemble des données d'entraînement et celui des données de test. Les données ont été séparées de manière à ce que les ensembles d'entraînement et de test ne partagent aucun nourrisson. Chaque enregistrement est classifié, en combinant les prédictions pour tous les échantillons lui appartenant. Ainsi, la probabilité que chaque enregistrement $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$ ait été produit par le modèle λ est obtenue en combinant les probabilités obtenues pour chacun des échantillons $\boldsymbol{x}_t$.

$$P(X|\lambda) = \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\lambda) \tag{4.1}$$

Ils ont évalué leurs performances avec les métriques EER (taux d'erreur égale) et AUC (aire sous la courbe), calculées à partir de la courbe DET (compromis de l'erreur de détection). Les résultats qu'ils ont obtenus sont présentés dans le tableau 4.2

Tableau 4.2 Performances obtenues par Farsaie Alaie *et al.* (2016) pour 2 durées de trames, 4 méthodes d'adaptation et deux version de LLR différentes

Trames	Adaptation	EER		AUC	
		LLR 1	LLR 2	LLR 1	LLR 2
10 ms	Méthode 1	0.2772	0.158415	0.810	0.932
	Méthode 2	0.2376	**0.148414**	0.840	**0.951**
	Méthode 3	0.2475	0.158415	0.826	**0.951**
	Méthode 4	0.2574	0.158415	0.827	0.932
30 ms	Méthode 1	0.2079	0.3267	0.870	0.760
	Méthode 2	0.1980	0.2772	0.890	0.825
	Méthode 3	0.2277	0.2475	0.860	0.824
	Méthode 4	0.1881	0.3069	0.890	0.770

Nous avons donc également produit la courbe DET de tous nos modèles, pour le *dataset* "référence", lorsque la seconde méthode de partition était employée. Nous avons utilisé la même équation que Farsaie Alaie *et al.* (2016), c'est-à-dire l'équation 4.1, pour ramener la classification au niveau des enregistrements. Les métriques EER et AUC obtenues, pour chacun des *folds*, pour chacun de nos modèles, sont présentées dans le tableau 4.3.

Tableau 4.3 Performances obtenues sur le *dataset* "référence", pour les 3 meilleurs modèles

Fold	Nombre de fichiers	EER			AUC		
		MLP	CNN	LSTM	MLP	CNN	LSTM
1	56	0.321	0.321	0.321	0.745	0.774	0.750
2	54	0.333	0.259	0.296	0.754	0.867	0.850
3	68	0.265	0.235	0.265	0.793	0.874	0.861
4	40	0.150	0.250	0.250	0.912	0.860	0.798
5	44	0.318	0.227	0.227	0.820	0.849	0.777
6	54	0.222	0.148	0.185	0.877	0.914	0.883
7	46	0.217	0.261	0.261	0.769	0.807	0.800
8	50	0.440	0.200	0.280	0.677	0.722	0.702
9	48	0.333	0.208	0.417	0.847	0.863	0.783
10	38	0.158	0.053	0.158	0.884	0.956	0.898
Moyenne	49.8	0.276	**0.216**	0.266	0.808	**0.849**	0.810
Écart-type	8.77	0.090	0.073	0.072	0.074	0.067	0.062

Les courbes DET obtenues pour chaque *fold*, pour chacun de nos modèles sont présentées en annexe III.

Les résultats que nous avons obtenus sont comparables avec la majorité de ceux de Farsaie Alaie *et al.* (2016). Leurs résultats avec la seconde version du LLR, pour des trames de 10 ms, sont cependant largement supérieurs aux nôtres.

Nous notons plusieurs différences importantes entre notre système et celui de Farsaie Alaie *et al.* (2016).

A. Premièrement, nous n'avons pas utilisé exactement le même *dataset* que Farsaie Alaie *et al.* (2016). En effet, nous avons simplement utilisé la même définition de classes qu'eux. Cependant, puisque leur article ne décrivait pas précisémment quelles pathologies étaient incluses dans le groupe *Other*, nous avons dû choisir. De plus, ils ont utilisé une version antérieure de la base de données, qui était plus petite ;

B. Deuxièmement, nous avons incorrectement partitionné nos données pour ce *dataset*. Les résultats présentés sont donc très peu fiables.

C. Troisièmement, il est important de mentionner que Farsaie Alaie *et al.* (2016) analysent les signaux avec une meilleure résolution temporelle que nous (et donc une résolution fréquentielle moindre). En effet, ils utilisent des trames de 10 ms ou 30 ms, avec 30% de recouvrement, tandis que nous utilisons des trames de 50 ms, sans recouvrement (comme l'ont fait Orozco-García & Reyes-García (2003b)) ;

D. Quatrièmement, l'extraction des MFCC est légèrement différente dans les deux études. Farsaie Alaie *et al.* (2016) utilisent une banque de 24 filtres mel et ne conservent que les 13 premiers coefficients, tandis que nous utilisons une banque de 20 filtres et conservons les 16 premiers coefficients (comme l'ont fait Orozco-García & Reyes-García (2003b)) ;

E. Cinquièmement, l'information contextuelle est représentée différemment dans les deux systèmes. Farsaie Alaie *et al.* (2016) modélisent les MFCC de chaque trame individuellement, mais ajoutent les dérivées première et seconde. De notre côté, nous modélisons les MFCC de séquences de 8 trames ;

F. Pour finir, Farsaie Alaie *et al.* (2016) utilisent des modèles GMM tandis que nous utilisons divers types de réseaux de neurones pour modéliser et classifier les attributs des trames.

Ainsi, cette comparaison ne permet aucune conclusion.

4.8 Réseaux de neurones

Au niveau théorique, nous savons que la force principale des réseaux de neurones est l'apprentissage de la représentation (*representation learning*). En effet, le design des attributs optimaux pour résoudre un problème est une tâche difficile. Lorsque beaucoup de données sont disponibles pour l'entraînement, de gros réseaux de neurones peuvent apprendre à extraire eux-mêmes l'information utile pour remplir la tâche qui leur est confiée (Goodfellow *et al.*, 2016).

Les réseaux de neurones ont aussi plusieurs désavantages, le principal étant la grande quantité de données et de ressources de calcul requis pour les entraîner. De plus, un niveau d'expertise très élevé est requis pour concevoir et entraîner ce type de modèle avec succès.

Dans le cadre de ce projet, la quantité limitée de données et de ressources de calculs n'a pas permis de tirer avantage de l'apprentissage de la représentation. De plus, malgré la faible taille de nos modèles, l'entraînement et l'optimisation des hyperparamètres ont requis beaucoup de temps et de calculs. En rétrospective, étant donné la quantité de données présentemment disponibles, il nous semble que ce type de modèle ne soit pas le plus approprié pour notre problème.

4.9 Attributs MFCC

Nous avons très peu discuté d'une étape cruciale dans n'importe quel algorithme d'apprentissage machine : le design des attributs. Nous avons utilisé dans ce projet les attributs MFCC suggérés par Orozco-García & Reyes-García (2003b).

Tel qu'expliqué à la section 3.4, les MFCC sont basés sur le spectrogramme de puissance mel, qui est calculé à partir du spectrogramme de puissance de la trame.

Rappelons que, dans le cadre de ce projet, nous utilisons des trames de 50 ms, ce qui est relativement long et signifie une bonne résolution fréquentielle (20 Hz), au coût d'une résolution temporelle passable. Nous n'employons aucun recouvrement entre les trames. De plus, seule la plage 0-4000 Hz est utilisée, même si notre fréquence d'échantillonnage permettrait en théorie d'étudier les signaux jusqu'à 22.5 kHz.

La figure 4.12 illustre un exemple de spectrogramme de puissance d'un échantillon. Le spectrogramme de puissance mel correspondant est également présenté.

On peut clairement observer la fréquence fondamentale ainsi que ses harmoniques dans le spectrogramme de puissance. On peut également observer la mélodie, bien que les transitions soient très abruptes d'une trame à l'autre en raison de la faible résolution temporelle et de

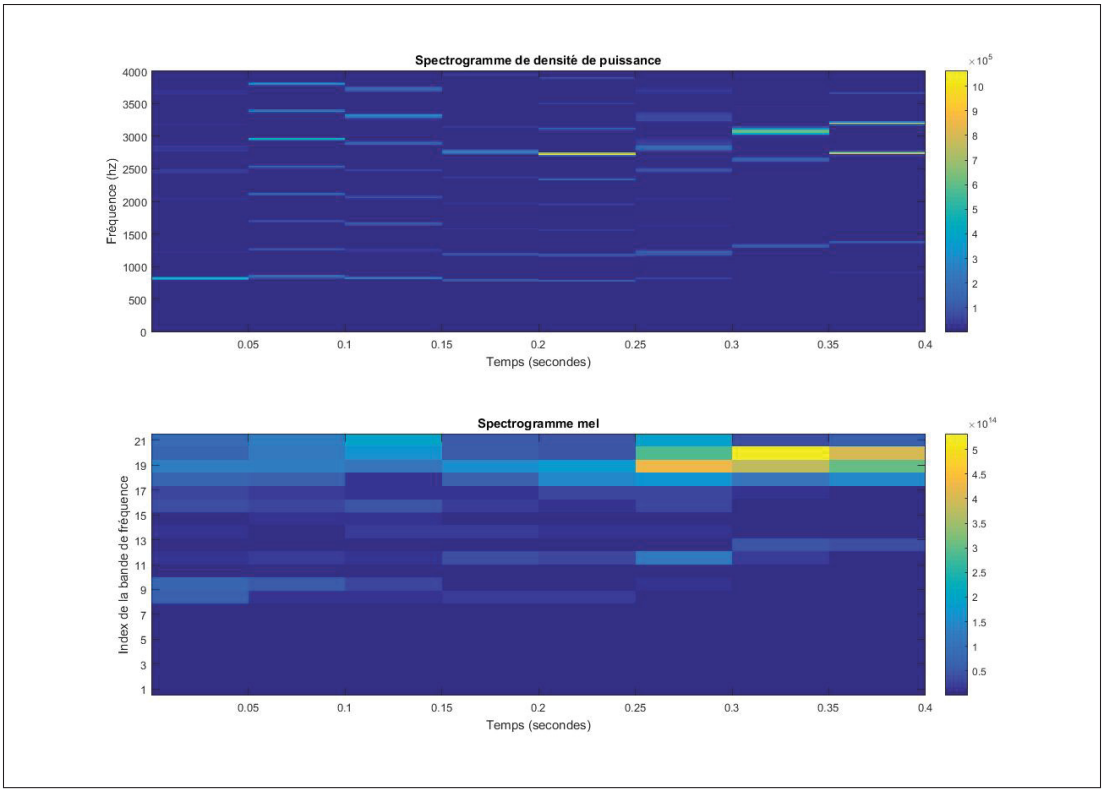

Figure 4.12 Spectrogramme de puissance (en haut) et spectrogramme de puissance mel (en bas) d'un échantillon, avec des trames de 50 ms et une recouvrement de 0% entre les trames. Une banque de 20 filtres mel a été utilisée

l'absence de recouvrement entre les trames. Une visualisation plus claire de ce spectrogramme est présentée en annexe IV-1.

L'application de la banque de 20 filtres mel vient réduire de beaucoup l'information contenue. En effet le nombre de points par trame passe de 201 à 20. De plus, l'espacement exponentiel entre les filtres vient déformer le spectrogramme. La fréquence fondamentale et ses harmoniques deviennent donc indiscernables. Des motifs importants, comme la mélodie ou la biphonation, ne peuvent plus être observés dans le spectrogramme de puissance mel. La transformée en cosinus discrète vient ensuite compresser l'information davantage pour produire les MFCC, qui sont utilisés comme attributs dans ce projet.

Nous doutons que ces attributs soient optimaux, car la majorité de l'information identifiée comme importante par plusieurs études pédiatriques est perdue. Les MFCC ne conservent pas

cette information, ils mettent plutôt en valeur la distribution spectrale de l'énergie. Il est fort possible que la perte de ces informations vienne limiter les performances de classification atteignables.

Toutes les études utilisant les MFCC comme attributs, y compris celle-ci, viennent néanmoins confirmer que les MFCC sont corrélés avec l'état de santé de l'enfant dont provient le cri.

Nous nous posons tout de même la question : La distribution spectrale de l'énergie des cris est-elle suffisante pour distinguer avec précision les cris d'enfants malades de ceux d'enfants en santé ? S'agit-il du meilleur type d'attributs possible pour cette tâche ?

D'ailleurs, selon notre revue de littérature, les performances atteintes par les études actuelles se basant sur les attributs MFCC sont loin d'être suffisantes pour la mise en marché, même lorsque chaque décision est basée sur l'analyse de plusieurs minutes d'enregistrement [1].

1. On ne tient pas compte ici des études menées par les groupes de recherche mexicains et malaisiens, car nous soupçonnons que leurs résultats soient biaisés (voir la section 4.4.1)

CONCLUSION ET RECOMMANDATIONS

Ainsi, dans le cadre de ce projet, nous avons travaillé sur le développement d'un système automatisé de dépistage par analyse de cris de nourrissons. Parmi tous les types de réseaux testés, il semble que les réseaux convolutionnels soient les mieux adaptés.

Nous n'avons cependant testé que de petits réseaux d'une couche cachée, en raison de la quantité limitée de données disponibles et de la quantité importante de calculs requis pour l'entraînement des réseaux et l'optimisation des hyperparamètres.

De plus, nos résultats semblent indiquer qu'en général, l'hyperbilirubinémie est plus facile à reconnaître que la détresse respiratoire. Difficile de comparer les autres pathologies étudiées, étant donné la faible quantité de données les représentant.

Nous avons également démontré l'importance d'une partition adéquate des données en ensembles d'entraînement et de validation, et avons remis en question les résultats de plusieurs études qui semblent avoir partitionné leurs données de façon inadéquate. Nous avons également réalisé que notre seconde méthode de partition était inadéquate lorsque la classe pathologique contenant plusieurs pathologies différentes. Ainsi, la validation croisée à *k folds* est difficile à appliquer pour cette tâche, car la partition des données doit respecter plusieurs contraintes. Une stratégie de validation basée sur la technique du *bootstrap* (Efron & Tibshirani, 1986) pourrait être plus appropriée pour cette tâche.

Pour la suite du projet, il nous semble important de concentrer les efforts sur la représentation. Nous croyons en effet que la perte d'information survenant lors de l'extraction des attributs vient limiter la performance atteignable.

Nous proposons de baser les attributs sur les symptômes spectrogrographiques identifiés dans plusieurs études pédiatriques. Nous pensons en particulier à l'analyse de la fréquence fondamentale et des formants, ainsi qu'au recensement de motifs comme la biphonation, la furcation,

les mélodies anormales, etc. L'extraction de ces attributs sera certainement difficile, mais pourrait grandement faciliter la classification.

Une autre approche intéressante à explorer serait l'apprentissage de la représentation. Cependant, une représentation apprise directement sur les *datasets* utilisés dans ce projet serait fort probablement inadéquate, car trop peu d'échantillons sont disponibles. Heureusement, des techniques comme le transfert d'apprentissage ou l'adaptation du domaine pourraient peut-être permettre de contourner ce problème.

ANNEXE I

CALCULS DES DÉRIVÉES ET JACOBIENS POUR L'EXEMPLE DE RÉTROPROPAGATION

Nous présentons ici, par soucis de complétude, le calcul des jacobiens de l'exemple de rétropropagation d'un MLP simple.

1. Gradient du coût par rapport à la sortie

Dans notre exemple, le coût est $J(\boldsymbol{\Theta}) = \mathscr{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_i^n y_i \ln \hat{y}_i$. Le gradient $\nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta})$ est donc :

$$\nabla_{\hat{\boldsymbol{y}}} J(\boldsymbol{\Theta}) = \begin{bmatrix} \frac{\partial J(\boldsymbol{\Theta})}{\partial \hat{\boldsymbol{y}}_1} \\ \frac{\partial J(\boldsymbol{\Theta})}{\partial \hat{\boldsymbol{y}}_2} \\ \vdots \\ \frac{\partial J(\boldsymbol{\Theta})}{\partial \hat{\boldsymbol{y}}_n} \end{bmatrix} = \begin{bmatrix} -\frac{y_1}{\hat{y}_1} \\ -\frac{y_2}{\hat{y}_2} \\ \vdots \\ -\frac{y_n}{\hat{y}_n} \end{bmatrix} \tag{A I-1}$$

2. Jacobien des activations

2.1 Jacobien de l'activation *softmax*

Dans cet exemple, nous utilisons une activation *softmax* pour la couche de sortie. Rappelons que l'activation *softmax* de $\boldsymbol{z}^{(L)}$ est :

$$\hat{y}_i = h_i^{(L)} = \text{softmax}(\boldsymbol{z})_i = \frac{\exp\left(z_i^{(L)}\right)}{\sum_k \exp\left(z_k^{(L)}\right)} \tag{A I-2}$$

Les éléments $\frac{\partial h_j^{(L)}}{\partial z_j^{(L)}}$ du jacobien $\frac{\partial \boldsymbol{h}^{(L)}}{\partial \boldsymbol{z}^{(L)}}$ valent donc :

$$\frac{\partial h_i^{(L)}}{\partial z_j^{(L)}} = \begin{cases} \frac{\exp(z_i)\sum_{k\neq i}\exp(z_k)}{\left(\sum_k \exp(z_k)\right)^2} = h_i^{(L)}\left(1-h_i^{(L)}\right) & i=j \\ -\frac{\exp\left(z_i+z_j\right)}{\left(\sum_k \exp(z_k)\right)^2} = -h_i^{(L)}h_j^{(L)} & i\neq j \end{cases} \tag{A I-3}$$

2.2 Jacobien de l'activation *ReLU*

Nous utilisons des activations *ReLU* pour les couches cachées dans cet exemple. Rappelons que l'activation *ReLU* de $\boldsymbol{z}^{(\ell)}$ est simplement :

$$h_i^{(\ell)} = \mathrm{ReLU}\left(z_i^{(\ell)}\right) = \max\left(z_i^{(\ell)}, 0\right) \tag{A I-4}$$

Les éléments $\frac{\partial h_i^{(\ell)}}{\partial z_j^{(\ell)}}$ du jacobien $\frac{\partial \boldsymbol{h}^{(\ell)}}{\partial \boldsymbol{z}^{(\ell)}}$ valent donc :

$$\frac{\partial h_i^{(\ell)}}{\partial z_j^{(\ell)}} = \begin{cases} 0 & i\neq j \\ \begin{cases} 0 & z_i^{(\ell)} <= 0 \\ 1 & z_i^{(\ell)} > 0 \end{cases} & i=j \end{cases} \tag{A I-5}$$

En réalité, la fonction *ReLU* est discontinue à l'origine, sa dérivée à ce point n'est donc pas définie. Afin de pouvoir calculer les gradients, on force artificiellement sa dérivée à être unitaire à l'origine.

3. Jacobien d'une transformation affine

3.1 Jacobien par rapport aux activations de la couche précédente

Puisque nous utilisons la transformation affine $\boldsymbol{z}^{(\ell)} = \boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell)} + \boldsymbol{b}^{(\ell)}$, les éléments $\frac{\partial z_i^{(\ell)}}{\partial h_j^{(\ell)}}$ de la matrice jacobienne $\frac{\partial \boldsymbol{z}^{(\ell)}}{\partial \boldsymbol{h}^{(\ell)}}$ sont :

$$\frac{\partial z_i^{(\ell)}}{\partial h_j^{(\ell)}} = W_{ij}^{(\ell)} \Rightarrow \frac{\partial \boldsymbol{z}^{(\ell)}}{\partial \boldsymbol{h}^{(\ell)}} = \boldsymbol{W}^{(\ell)} \tag{A I-6}$$

3.2 Jacobien par rapport aux paramètres

Pour finir, les dérivées partielles $\frac{\partial z_i^{(\ell)}}{\partial \theta_j^{(\ell)}}$ de la matrice jacobienne $\frac{\partial \boldsymbol{z}^{(\ell)}}{\partial \boldsymbol{\theta}^{(\ell)}}$ sont calculées ainsi

Si le paramètre $\theta_j^{(\ell)}$ est un poid $W_{kl}^{(\ell)}$, alors $\frac{\partial z_i^{(\ell)}}{\partial W_{kl}^{(\ell)}} = \begin{cases} h_l^{(\ell)} & k = i \\ 0 & m \neq i \end{cases}$

Si le paramètre $\theta_j^{(\ell)}$ est un biais $b_k^{(\ell)}$, alors $\frac{\partial z_i^{(\ell)}}{\partial b_k^{(\ell)}} = \begin{cases} 1 & k = i \\ 0 & k \neq i \end{cases}$

ANNEXE II

PARTITION DES DONNÉES

1. Partition par la première méthode

La table II-1 résume la partition des données de chacun des *datasets* par la première méthode de partition.

Tableau-A II-1 Partition des données en *folds* par la première méthode

Nom du *dataset*		Disponibles		Nombre d'échantillons dans chacun des *folds*									
		Bébés	Éch.	1	2	3	4	5	6	7	8	9	10
Asphyxie	Classe 1	305	75284	37	37	37	37	37	37	37	37	37	37
	Classe 2	1	370	37	37	37	37	37	37	37	37	37	37
Syndrome de Down	Classe 1	305	75284	18	18	18	17	17	17	17	17	17	17
	Classe 2	1	173	18	18	18	17	17	17	17	17	17	17
Hyperbilirubinémie	Classe 1	305	75284	188	188	188	188	188	188	188	188	188	187
	Classe 2	7	1879	188	188	188	188	188	188	188	188	188	187
Hypoglycémie	Classe 1	305	75284	91	91	91	91	91	91	90	90	90	90
	Classe 2	3	906	91	91	91	91	91	91	90	90	90	90
Méningite	Classe 1	305	75284	50	50	50	50	50	50	50	50	50	49
	Classe 2	2	499	50	50	50	50	50	50	50	50	50	49
Dépistage	Classe 1	305	75284	383	383	383	383	383	383	383	382	382	382
	Classe 2	14	3827	383	383	383	383	383	383	383	382	382	382
Détresse respiratoire	Classe 1	305	75284	941	941	941	941	941	941	941	941	941	940
	Classe 2	35	9409	941	941	941	941	941	941	941	941	941	940
Référence	Classe 1	305	75284	2275	2275	2274	2274	2274	2274	2274	2274	2274	2274
	Classe 2	84	22742	2275	2275	2274	2274	2274	2274	2274	2274	2274	2274

2. Partition par la seconde méthode

La table II-2 résume la partition des données de chacun des *datasets* par la seconde méthode de partition.

Tableau-A II-2 Partition des données en *folds* par la seconde méthode

Nom du *dataset*		Disponibles		Fold 1		Fold 2		Fold 3		Fold 4		Fold 5		Fold 6		Fold 7		Fold 8		Fold 9		Fold 10	
		Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.	Bébés	Échs.
Hyperbilirubinémie	Classe 1	305	75284	37	552	29	77	37	95	38	379	40	191	50	334	41	251	-	-	-	-	-	-
	Classe 2	7	1879	1	552	1	77	1	95	1	379	1	191	1	334	1	251	-	-	-	-	-	-
Hypoglycémie	Classe 1	305	75284	90	439	87	247	73	220	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Classe 2	3	906	1	439	1	247	1	220	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Méningite	Classe 1	305	75284	106	239	111	260	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Classe 2	2	499	1	239	1	260	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Dépistage	Classe 1	305	75284	28	430	29	552	31	334	26	379	31	393	28	534	31	324	32	260	33	251	27	370
	Classe 2	14	3827	2	430	1	552	1	334	1	379	2	393	2	534	2	324	1	260	1	251	1	370
Détresse respiratoire	Classe 1	305	75284	28	794	29	903	31	841	26	889	33	1012	29	1153	31	861	33	778	34	930	27	1248
	Classe 2	35	9409	4	794	4	903	3	841	2	889	3	1012	4	1153	3	861	4	778	4	930	4	1248
Référence	Classe 1	305	75284	28	2521	29	2236	31	2326	26	2256	33	2188	29	2111	31	2322	33	2288	34	2269	27	2225
	Classe 2	84	22742	9	2521	7	2236	9	2326	10	2256	8	2188	7	2111	12	2322	6	2288	7	2269	9	2225

ANNEXE III

COURBES DET

1. Modèle MLP

La figure III-1 présente les courbes DET obtenues pour chacun des *folds* sur le *dataset* "référence", avec le modèle MLP.

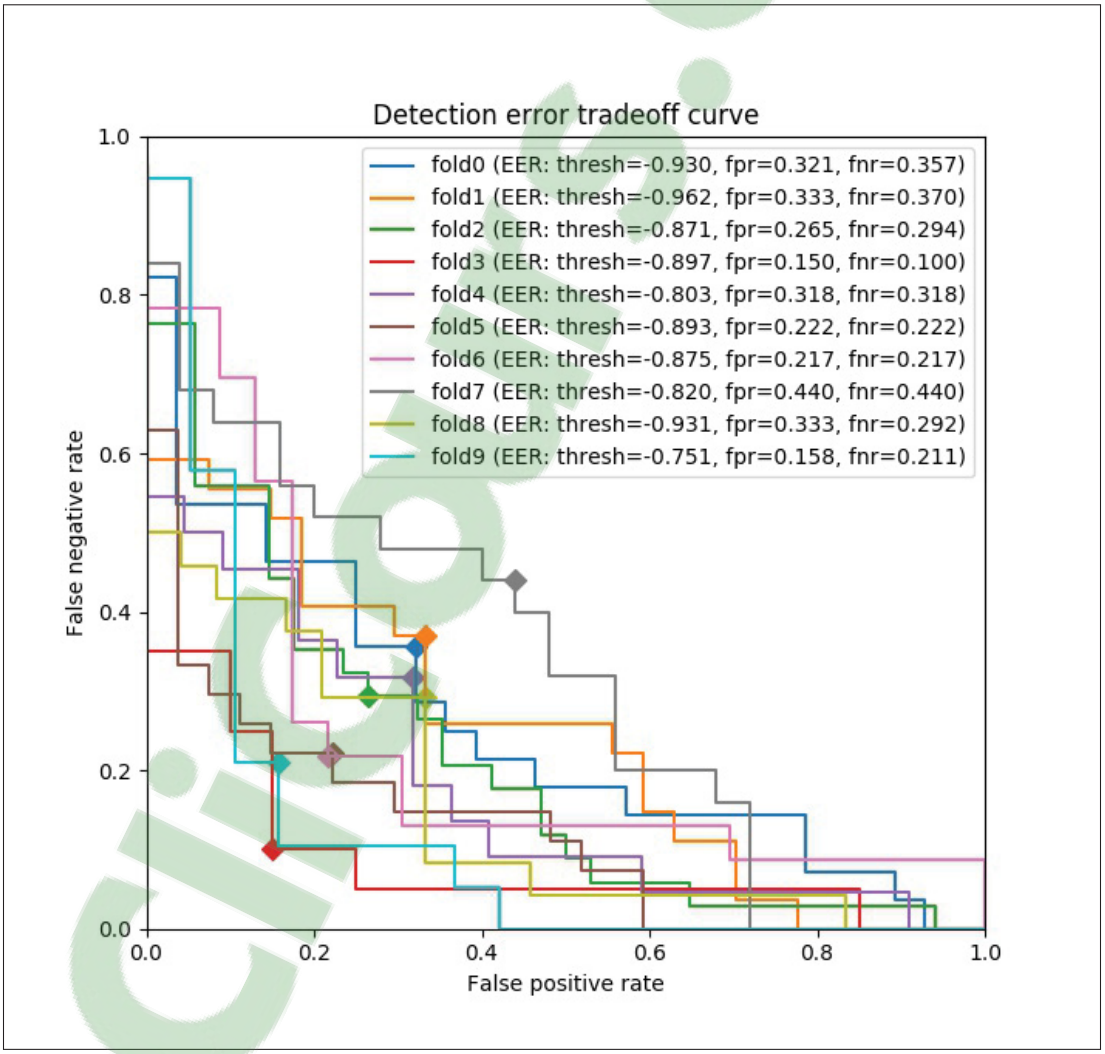

Figure-A III-1 Courbe DET de chacun des *folds* du modèle MLP sur le *dataset* "référence"

2. Modèle CNN

La figure III-2 présente les courbes DET obtenues pour chacun des *folds* sur le *dataset* "référence", avec le modèle CNN.

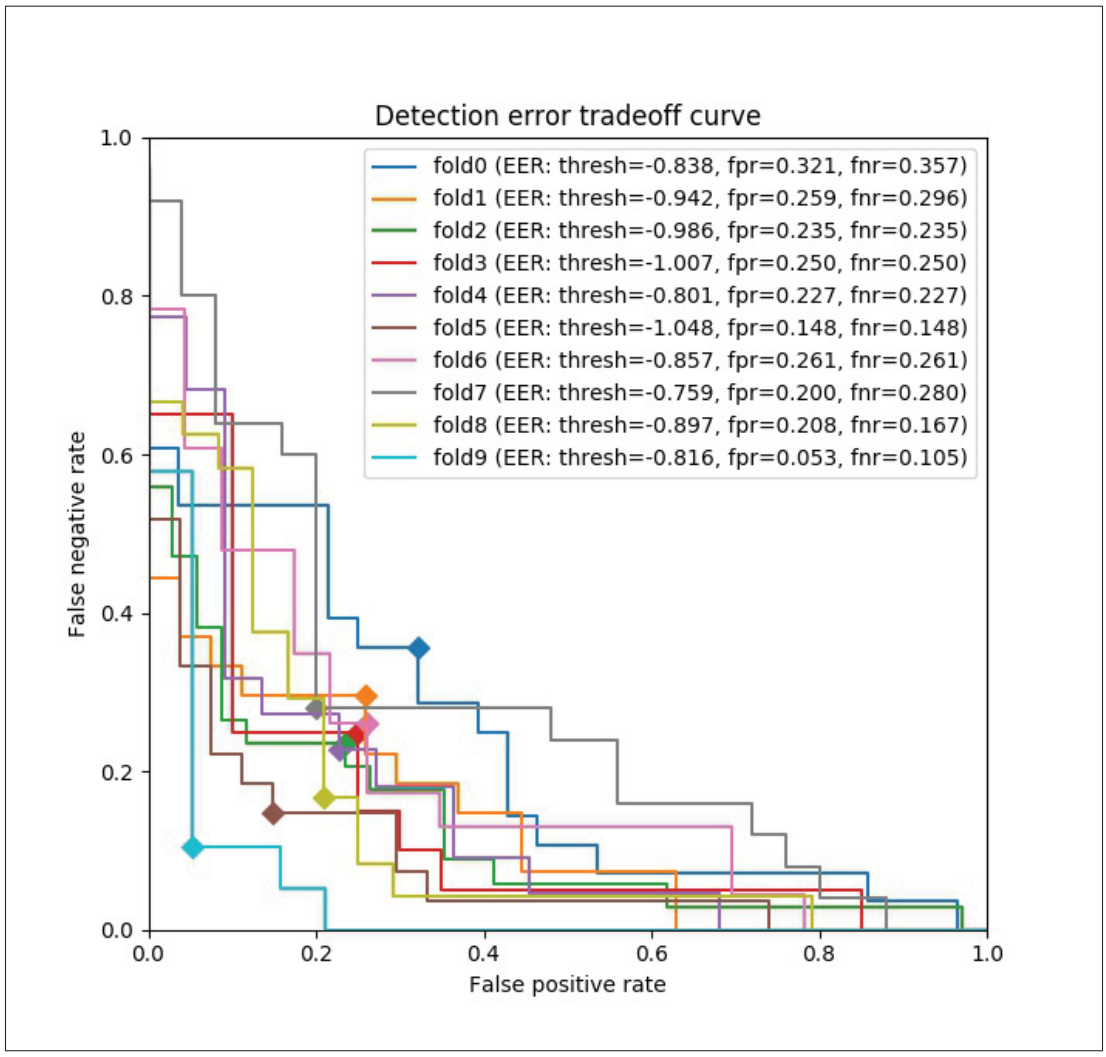

Figure-A III-2 Courbe DET de chacun des *folds* du modèle CNN sur le *dataset* "référence"

3. Modèle LSTM

La figure III-3 présente les courbes DET obtenues pour chacun des *folds* sur le *dataset* "référence", avec le modèle LSTM.

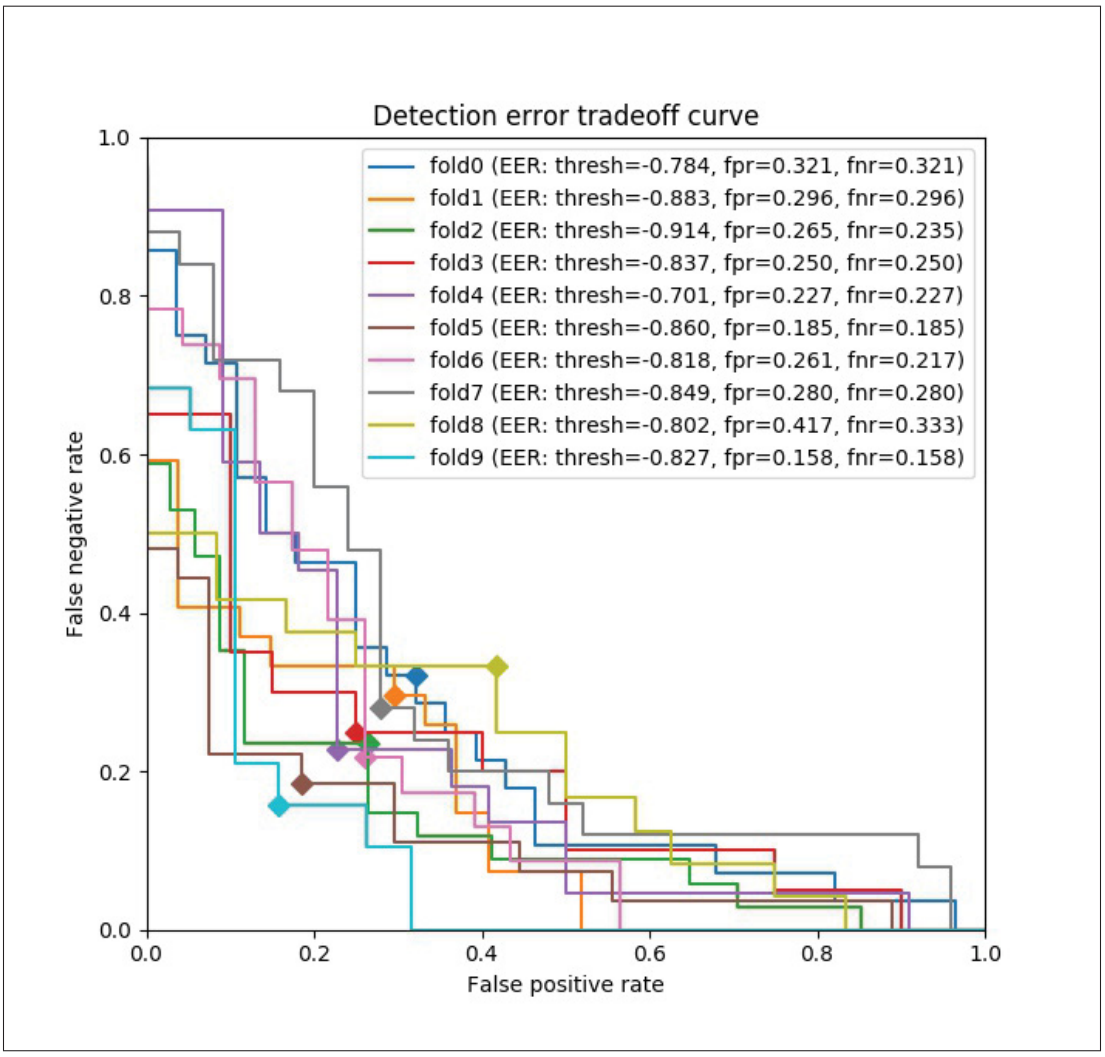

Figure-A III-3 Courbe DET de chacun des *folds* du modèle LSTM sur le *dataset* "référence"

ANNEXE IV

EXEMPLE DE SPECTROGRAMME DE PUISSANCE

Un exemple de spectrogramme de puissance produit lors du calcul des MFCC a été présenté à la figure 4.12.

La figure IV-1 présente le spectrogramme de puissance du même signal, avec une résolution temporelle plus élevée, afin d'améliorer la lisibilité. Le spectrogramme de puissance mel correspondant est également présenté.

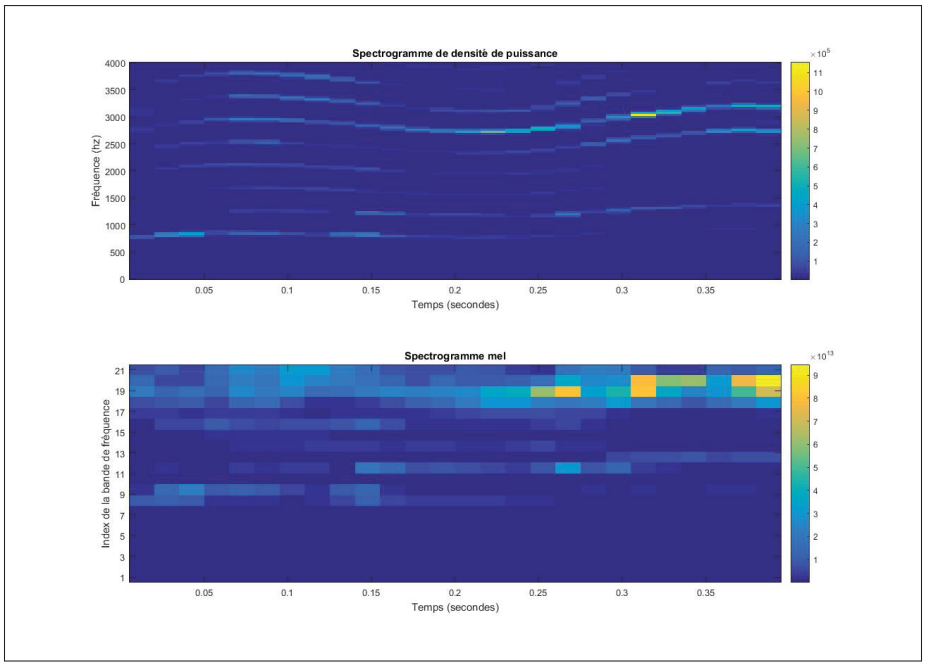

Figure-A IV-1 Spectrogramme de puissance (en haut) et spectrogramme de puissance mel (en bas) d'un échantillon, avec des trames de 25 ms et un recouvrement de 40% entre les trames. Une banque de 20 filtres mel a été utilisée.

BIBLIOGRAPHIE

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015). *TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems*. arXiv preprint. doi : 10.1038/nn.3331.

Abou-abbas, L., Farsaie Alaie, H. & Tadj, C. (2015a). Automatic detection of the expiratory and inspiratory phases in newborn cry signals. *Biomedical signal processing and control*, 19, 35–43. doi : 10.1016/j.bspc.2015.03.007.

Abou-abbas, L., Farsaie Alaie, H. & Tadj, C. (2015b). Segmentation of voiced newborns' cry sounds using Wavelet Packet Based Features. *2015 ieee 28th canadian conference on electrical and computer engineering (ccece)*, pp. 27–40.

Abou-abbas, L., Montazeri, L., Gargour, C. & Tadj, C. (2015c). On the Use of EMD for Automatic Newborn Cry Segmentation. pp. 262–265.

Abou-abbas, L., Tadj, C., Gargour, C. & Montazeri, L. (2016). Expiratory and Inspiratory Cries Detection Using Different Signals ' Decomposition Techniques. *Journal of voice*.

Antonucci, R., Porcella, A. & Pilloni, M. D. (2014). Perinatal asphyxia in the term newborn. *Journal of pediatric and neonatal individualized medicine*, 3(2), 1–14. doi : 10.7363/030269.

Barajas-Montiel, S. E. & Reyes-García, C. A. (2006). Fuzzy support vector machines for automatic infant cry recognition. *Lecture notes in control and information sciences*, 345, 876–881. doi : 10.1007/11816515{_}107.

Bell, S. M. & Salter Ainsworth, M. D. (1972). Infant Crying and Maternal Responsiveness. *Child development*, 43(4), 1171–1190.

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning Long Term Dependencies with Gradient Descent is Difficult. *Ieee transactions on neural networks*, 5(2), 157–166. doi : 10.1109/72.279181.

Bergstra, J. & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of machine learning research*, 13, 281–305. doi : 10.1162/153244303322533223.

Boersma, P. & van Heuven, V. (2001). PRAAT, a system for doing phonetics by computer. *Glot international*, 5(9-10), 341–347. doi : 10.1097/AUD.0b013e31821473f7.

Bogert, B. P., Healy, J. R. & Tukey, J. W. (1963). The Quefrency Analysis of Time Series for Echoes : Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum, and Saphe Cracking. *Symposium on time series analysis*, pp. 209–243.

Bos, A. F., Van Braeckel, K. N. J. A., Hitzert, M. M., Tanis, J. C. & Roze, E. (2013). Development of fine motor skills in preterm infants. *Developmental medicine & child neurology*, 55, 1–4. doi : 10.1111/dmcn.12297.

Bottou, L., Curtis, F. E. & Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. *Society for industrial and applied mathematics*, 60(2), 223–311. Repéré à http://arxiv.org/abs/1606.04838.

Chittora, A. & Patil, H. A. (2015). Classification of normal and pathological infant cries using bispectrum features. *2015 23rd european signal processing conference, eusipco 2015*, 639–643. doi : 10.1109/EUSIPCO.2015.7362461.

Chollet, F. & Others. (2015). Keras. Repéré à https://keras.io.

Clement, C. J. & Beinum, F. J. K. V. (1995). Influence of lack of auditory feedback : vocalizations of deaf and hearing infants compared. *Proceedings of the institute of phonetic sciences*, 19, 25–37. Repéré à http://uvafon.hum.uva.nl/Proceedings/Proceedings_19/ChrisClement/ChrisClement.ps.

Colton, R. H. & Steinschneider, A. (1981). The cry characteristics of an infant who died of the sudden infant death syndrome. *The journal of speech and hearing disorders*, 46(4), 359–363.

Corwin, M., Lester, B. M., Sepkoski, C., Peucker, M., Kayne, H. & Golub, H. (1995). Newborn acoustic cry characteristics of infants subsequently dying of sudden infant death syndrome. *Pediatrics*, 96(1), 73–77.

Cullen, J. K., Fargo, N., Chase, R. A. & Baker, P. (1967). The development of auditory feedback monitoring : I. Delayed auditory feedback studies on infant cry. *Journal of speech, language, and hearing research*, 11(1), 85–93.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2, 303–314. doi : 10.1007/BF02836480.

Davis, S. B. & Mermelstein, P. (1980). *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*. IEEE Transactions on Acoustics, Speech, and Signal Processing. doi : 10.1109/TASSP.1980.1163420.

Efron, B. & Tibshirani, R. (1986). Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical science*, 1(1), 54–77. doi : 10.1016/j.csda.2010.05.004.

Eilers, R. E. & Oller, D. K. (1994). Infant vocalizations and the early diagnosis of severe hearing impairment. *The journal of pediatrics*, 124(2), 199–203.

Ellis, D. P. W. (2005). PLP and RASTA (and MFCC, and inversion) in Matlab. Repéré à http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/.

Farsaie Alaie, H. & Tadj, C. (2012). Cry-based classification of healthy and sick infants using adapted boosting mixture learning method for gaussian mixture models. *Modelling and simulation in engineering*, 2012, 5–9. doi : 10.1155/2012/983147.

Farsaie Alaie, H. & Tadj, C. (2013). Splitting of Gaussian Models via Adapted BML Method Pertaining to Cry-Based Diagnostic System. *Engineering*, 5, 277–283.

Farsaie Alaie, H., Abou-Abbas, L. & Tadj, C. (2016). Cry-based infant pathology classification using GMMs. *Speech communication*, 77, 28–52. doi : 10.1016/j.specom.2015.12.001.

Fisichelli, V. R. & Karelitz, S. (1963). The cry latencies of normal infants and those with brain damage. *The journal of pediatrics*, 62(5), 724–734.

Fraser, J., Walls, M. & McGuire, W. (2004). Respiratory complications of preterm birth. *Bmj (clinical research ed.)*, 329(7472), 962–965. doi : 10.1136/bmj.329.7472.962.

Goldenberg, R. L., Culhane, J. F., Iams, J. D. & Romero, R. (2008). Epidemiology and causes of preterm birth. *The lancet*, 371(9606), 75–84. doi : 10.1016/S0140-6736(08)60074-4.

Golub, H. & Corwin, M. (1985). A Physioacoustic Model of the Infant Cry. Dans Lester, B. M. & Zachariah Boukydis, C. F. (Éds.), *Infant Crying : Theoretical and Research Perspectives* (ch. 3, pp. 59–82). Boston, MA : Plenum Press.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. 1–43. doi : 10.1145/2661829.2661935.

Gustafsson, E., Levréro, F., Reby, D. & Mathevon, N. (2013). Fathers are just as good as mothers at recognizing the cries of their baby. *Nature communications*, 4, 1698. doi : 10.1038/ncomms2713.

Hariharan, M., Yaacob, S. & Awang, S. A. (2011). Pathological infant cry analysis using wavelet packet transform and probabilistic neural network. *Expert systems with applications*, 38(12), 15377–15382. doi : 10.1016/j.eswa.2011.06.025.

Hariharan, M., Chee, L. S. & Yaacob, S. (2012a). Analysis of infant cry through weighted linear prediction cepstral coefficients and probabilistic neural network. *Journal of medical systems*, 36(3), 1309–1315. doi : 10.1007/s10916-010-9591-z.

Hariharan, M., Sindhu, R. & Yaacob, S. (2012b). Normal and hypoacoustic infant cry signal classification using time-frequency analysis and general regression neural network. *Computer methods and programs in biomedicine*, 108(2), 559–569. doi : 10.1016/j.cmpb.2011.07.010.

Hochreiter, S., Bengio, Y. & Frasconi, P. (2001). Gradient Flow in Recurrent Nets : The Difficulty of Learning LongTerm Dependencies. *A field guide to dynamical recurrent networks*. doi : 10.1109/9780470544037.ch14.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2), 251–257. doi : 10.1016/0893-6080(91)90009-T.

Hu, J., Shen, L. & Sun, G. (2017). *Squeeze-and-Excitation Networks*. Repéré à http://arxiv.org/abs/1709.01507.

Ioffe, S. & Szegedy, C. (2015). Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Arxiv*, 1–11. doi : 10.1007/s13398-014-0173-7.2.

Johnson, B. (2017). Learning Disabilities in Children : Epidemiology, Risk Factors and Importance of Early Intervention. *Bmh medical journal - issn 2348–392x*, 4(1), 31–37. Repéré à http://www.babymhospital.org/BMH_MJ/index.php/BMHMJ/article/view/120.

Jost, K., Pramana, I., Delgado-Eckert, E., Kumar, N., Datta, A. N., Frey, U. & Schulzke, S. M. (2017). Dynamics and complexity of body temperature in preterm infants nursed in incubators. *Plos one*, 12(4), 1–15. doi : 10.1371/journal.pone.0176670.

Juang, B. H., Rabiner, L. R. & Wilpon, J. G. (1987). On the use of bandpass filtering in speech recognition. *Ieee transactions on acoustics, speech, and signal processing*, 35(7), 947–954.

Juntunen, K., Sirviö, P. & Michelsson, K. (1978). Cry analysis in infants with severe malnutrition. *European journal of pediatrics*, 128(4), 241–246.

Kim, K. S. (2010). Acute bacterial meningitis in infants and children. *The lancet infectious diseases*, 10(1), 32–42. doi : 10.1016/S1473-3099(09)70306-8.

Kingma, D. P. & Ba, J. L. (2015). Adam : a Method for Stochastic Optimization. *International conference on learning representations 2015*, 1–15.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Appears in the international joint conference on articial intelligence (ijcai)*, 5, 1–7. doi : 10.1067/mod.2000.109031.

Koivisto, M., Michelsson, K., Sirviö, P. & Wasz-Höckert, O. (1974). Spectrographic analysis of pain cry of hypoglycemia in newborn infants. *Xiv international congress of pediatrics*, pp. 250.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, 1–9. doi : http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Lieberman, P. (1985). The Physiology of Cry and Speech in Relation to Linguistic Behavior. Dans Lester, B. M. & Zachariah Boukydis, C. F. (Éds.), *Infant Crying : Theoretical and Research Perspectives* (ch. 2, pp. 29–57). Boston, MA : Plenum Press.

Lind, J., Vuorenkoski, V., Rosberg, G., Partanen, T. J. & Wasz-Höckert, O. (1970). Spectographic Analysis of Vocal Response to Pain Stimuli in Infants with Down's Syndrome. *Developmental medicine & child neurology*, 12(4), 478–486.

Luchsinger, R., Dubois, C., Vassela, F., Joss, E., Gloor, R. & Wiesmann, U. (1967). Spektralanalyse des "Miauens" bei Cri-du-Chat Syndrom. *Folia phoniatrica*, 19, 27–33.

McGuire, W., Clerihew, L. & Fowlie, P. W. (2004). Infection in the preterm infant. *Bmj (clinical research ed.)*, 329(7477), 1277–80. doi : 10.1136/bmj.329.7477.1277.

Michelsson, K. (1971). Cry analysis of symptomless low birth weight neonates and of asphyxiated newborn infants. *Acta paediatrica scandinavica*, 216, 1–45.

Michelsson, K. & Sirviö, P. (1976). Cry Analysis in Congenital Hypothyroidism. *Folia phoniatrica et logopaedica*, 28(1), 40–47.

Michelsson, K., Sirviö, P. & Wasz-Höckert, O. (1977a). Pain cry in full-term asphyxiated newborn infants correlated with late findings. *Acta paediatrica scandinavica*, 66(5), 611–616.

Michelsson, K., Sirviö, P. & Wasz-Höckert, O. (1977b). Sound spectrographic cry analysis of infants with bacterial meningitis. *Developmental medicine & child neurology*, 19, 309–315.

Michelsson, K., Tuppurainen, N. & Aula, P. (1980). Cry analysis of infants with karyotype abnormality. *Neuropediatrics*, 11(4), 365–376.

Michelsson, K., Raes, J., Thodén, C.-J. & Wasz-Höckert, O. (1982). Sound spectrographic cry analysis in neonatal diagnostics : An evaluative study. *Journal of phonetics*, 10(1), 79–88.

Michelsson, K., Kaskinen, H., Aulanko, R. & Rinne, A. (1984). Sound spectrographic cry analysis of infants with hydrocephalus. *Acta paediatrica scandinavica*, 73(1), 65–68.

Mitchell, T. M. (1997). *Machine Learning*. New-York.

Möller, S. & Schönweiler, R. (1999). Analysis of infant cries for the early detection of hearing impairment. *Speech communication*, 28(3), 175–193. doi : 10.1016/S0167-6393(99)00016-3.

Morsbach, G. & Bunting, C. (1979). Maternal Recognition of their Neonates' Cries. *developmental medicine & child neurology*, 21(2), 178–185.

Nissen, M. D. (2007). Congenital and neonatal pneumonia. 8(3), 195–203. doi : 10.1016/j.prrv.2007.07.001.

Noll, M. A. (1967). Cepstrum Pitch Determination. *The journal of the acoustical society of america*, 41(2), 293–309.

Oppenheim, A. V. & Schafer, R. W. (2004). From frequency to quefrency : A history of the cepstrum. *Ieee signal processing magazine*, 21(5), 95–106. doi : 10.1109/MSP.2004.1328092.

Orozco-García, J. & Reyes-García, C. A. (2003a). A study on the Recognition of Patterns of Infant Cry for the Identification of Deafness in Just Born Babies with neural Networks. *Progress in pattern recognition, speech and image analysis*, (January 2016), 342–349. doi : 10.1007/978-3-540-24586-5{_}42.

Orozco-García, J. & Reyes-García, C. A. (2003b). Acoustic Features Analysis for Recognition of Normal and Hypoacoustic Infant Cry Based on Neural Networks. *Artificial neural nets problem solving methods*, 2687(January 2003), 615–622. Repéré à http://www.springerlink.com/content/4pn152frr42qvmhr/.

O'Shaughnessy, D. (1987). *Speech communication : human and machine*.

Ostwald, P. F., Phibbs, R. & Fox, S. (1968). Diagnostic use of infant cry. *Biology of the neonate*, 13(1), 68–82.

Petroni, M., Malowany, A. S., Johnston, C. C. & Stevens, B. J. (1995). Classification of infant cry vocalizations using artificial neural networks (ANNs). *1995 international conference on acoustics, speech, and signal processing, 1995. icassp-95*, (Cim), 3475–3478.

Pettay, O., Donner, M., Michelsson, K. & Sirviö, P. (1977). New aspects on the diagnosis of herpes simplex virus (HSV) infections in the newborn. *Xv international congress of pediatrics*, pp. 235.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5), 1–17.

Python Software Foundation. (2018). Python Language Reference, version 3.5. Repéré à http://www.python.org.

Rabiner, L. R. & Juang, B.-H. (1993). *Fundamentals of Speech Recognition*.

Reyes-Galaviz, O. F. & Reyes-García, C. A. (2004). Infant Cry Classification to Identify Hypoacoustics and Asphyxia with Neural Networks. *Micai 2004 : Advances in artificial intelligence : Third mexican international conference on artificial intelligence, mexico city, mexico, april 26-30, 2004. proceedings*, 69–78. doi : 10.1007/978-3-540-24694-7{_}8.

Reyes-Galaviz, O. F., Verduzco-Mendoza, A., Arch-tirado, E. & Reyes-García, C. A. (2005). Analysis of an Infant Cry Recognizer for the Early Identification of Pathologies. *Lecture notes in computer science*, 3445, 404–409. doi : 10.1007/b138975.

Reyes-Galaviz, O. F., Cano-Ortiz, S. D. & Reyes-García, C. A. (2008). Evolutionary-neural system to classify infant cry units for pathologies identification in recently born babies. *7th mexican international conference on artificial intelligence - proceedings of the special session, micai 2008*, 330–335. doi : 10.1109/MICAI.2008.73.

Reynolds, D. A. (2008). Gaussian Mixture Models. *Encyclopedia of biometric recognition*, 31(2), 1047–64. doi : 10.1088/0967-3334/31/7/013.

Robb, M. P., Crowell, D. H. & Dunn-Rankin, P. (2013). Sudden Infant Death Syndrome : Cry characteristics. *International journal of pediatric otorhinolaryngology*, 77(8), 1263–1267. doi : 10.1016/j.ijporl.2013.05.005.

Rosales-Pérez, A., Reyes-García, C. A. & Gómez-Gil, P. (2011). Genetic fuzzy relational neural network for infant cry classification. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, 6718 LNCS, 288–296. doi : 10.1007/978-3-642-21587-2{_}31.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533. Repéré à https://doi.org/10.1038/323533a010.1038/323533a0.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International journal of computer vision*, 115(3), 211–252. doi : 10.1007/s11263-015-0816-y.

Sahak, R., Mansor, W., Khuan, L. Y., Zabidi, A. & Yassin, A. I. (2012). Detection of asphyxia from infant cry using support vector machine and multilayer perceptron integrated with Orthogonal Least Square. *Proceedings - ieee-embs international conference on biomedical and health informatics : Global grand challenge of health informatics, bhi 2012*, 25(Bhi), 906–909. doi : 10.1109/BHI.2012.6211734.

Sahak, R., Mansor, W., Khuan, L. Y., Mohd Yassin, A. I. & Zabidi, A. (2010). Orthogonal least square based support vector machine for the classification of infant cry with asphyxia. *Proceedings - 2010 3rd international conference on biomedical engineering and informatics, bmei 2010*, 3(Bmei), 986–990. doi : 10.1109/BMEI.2010.5639300.

Santiago-Sánchez, K., Reyes-García, C. A. & Gómez-Gil, P. (2009). Type-2 fuzzy sets applied to pattern matching for the classification of cries of infants under neurological risk. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, 5754 LNCS(1), 201–210. doi : 10.1007/978-3-642-04070-2{_}23.

Saraswathy, J., Hariharan, M., Khairunizam, W., Yaacob, S. & Thiyagar, N. (2013). Infant cry classification : Time frequency analysis. *Proceedings - 2013 ieee international conference on control system, computing and engineering, iccsce 2013*, 499–504. doi : 10.1109/ICCSCE.2013.6720016.

Shane, A. L., Sánchez, P. J. & Stoll, B. J. (2017). Neonatal sepsis. *The lancet*, 390(10104), 1770–1780. doi : 10.1016/S0140-6736(17)31002-4.

Sjölander, K. & Beskow, J. (2000). Wavesurfer – an open source speech tool. *Interspeech*, 4(Icslp), 464–467.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of machine learning research*, 15, 1929–1958. doi : 10.1214/12-AOS1000.

Stevens, S. S., Volkmann, J. & Newman, E. B. (1937). A Scale for the Measurement of the Psychological Magnitu Pitch. *The journal of the acoustical society of america*, 8(3), 185–190. doi : 10.1121/1.1915893.

Suaste-Rivas, I., Reyes-Galaviz, O. F., Diaz-Mendez, A. & Reyes-García, C. A. (2004a). A Fuzzy Relational Neural Network for Pattern Classification. *Progress in pattern recognition, image analysis and applications : 9th iberoamerican congress on pattern recognition, ciarp 2004, puebla, mexico, october 26-29, 2004. proceedings*, 358–365. doi : 10.1007/978-3-540-30463-0{_}44.

Suaste-Rivas, I., Reyes-Galaviz, O. F., Diaz-Mendez, A. & Reyes-García, C. A. (2004b). Implementation of a linguistic fuzzy relational neural network for detecting pathologies by infant cry recognition. *Advances in artificial intelligence – iberamia 2004*, 953–962. doi : 10.1007/978-3-540-30498-2{_}95.

Sutskever, I., Vinyals, O. & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. 1–9. doi : 10.1007/s10107-014-0839-0.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the ieee computer society conference on computer vision and pattern recognition*, 07-12-June, 1–9. doi : 10.1109/CVPR.2015.7298594.

The MathWorks Inc. (2016). *Matlab Release 2016a*. Massachusetts, United States. Natick, Massachusetts, United States. doi : 2016-11-26.

Thodén, C.-J. & Koivisto, M. (1980). Acoustic analysis of the normal pain cry. *Infant communication : Cry and early speech*, 124–151.

Thodén, C.-J. & Michelsson, K. (1979). Sound spectrographic cry analysis in Krabbe's disease. *Developmental medicine & child neurology*, 21(3), 400–402.

Thodén, C.-J., Jarvenpaa, A.-L. & Michelsson, K. (1985). Sound Spectrographic Cry Analysis of Pain Cry in Prematures. Dans Lester, B. M. & Zachariah Boukydis, C. F. (Éds.), *Infant Crying : Theoretical and Research Perspectives* (ch. 5, pp. 105–118). Boston, MA.

Thwaites, C. L., Beeching, N. J. & Newton, C. R. (2015). Maternal and neonatal tetanus. *The lancet*, 385(9965), 362–370. doi : 10.1016/S0140-6736(14)60236-1.

Unicef & World Health Organization. (2015). *A decade of tracking progress for maternal, newborn, and child survival : 2015 Report.* (Rapport n°15).

Unicef, World Health Organization, World Bank Group & United Nations. (2017). *Levels & Trends in Child Mortality*.

United Nations. (2000). United Nations Millenium Declaration.

United Nations. (2015). Transforming our world : the 2030 Agenda for Sustainable Development. (October), 1–35. doi : 10.1007/s13398-014-0173-7.2.

Valanne, E., Vuorenkoski, V., Partanen, T., Lind, J. & Wasz-Höckert, O. (1967). The ability of human mothers to identify the hunger cry signals of their own new-born infants during the lying-in period. *Experientia*, 23(9), 768–769.

Volkmar, F. R. (2014). Editorial : The Importance of Early Intervention. *Journal of autism and developmental disorders*, 44(12), 2979–2980. doi : 10.1007/s10803-014-2265-9.

Vuorenkoski, V., Lind, J., Partanen, T. J., Lejeune, J., Lafourcade, J. & Wasz-Höckert, O. (1966). Spectrographic analysis of cries from children with maladie du cri du chat. *Annales paediatriae fenniae*, 12(3), 174–180.

Wahid, N. S. A., Saad, P. & Hariharan, M. (2016). Infant Cry Classification Using Radial Basis Function Network. *Journal of advanced research in applied sciences and engineering technology*, 4(1), 12–28.

Wasz-Höckert, O., Lind, J., Vuorenkoski, V., Partanen, T. J. & Valanne, E. (1968). *The Infant cry : a spectrographic and auditory analysis*.

Wasz-Höckert, O., Koivisto, M., Vuorenkoski, V., Partanen, T. J. & Lind, J. (1971). Spectrographic analysis of pain cry in hyperbilirubinemia. *Biology of the neonate*, 17, 260–271.

Wasz-Höckert, O., Michelsson, K. & Lind, J. (1985). Twenty-Five Years of Scandinavian Cry Research. Dans Lester, B. M. & Zachariah Boukydis, C. F. (Éds.), *Infant Crying : Theoretical and Research Perspectives* (ch. 4, pp. 83–104). Boston, MA : Plenum Press. doi : 10.1007/978-1-4613-2381-5{_}4.

Wolf, P. (1969). The natural history of crying and other vocalization in early infancy. Dans *Determinants of Infant Behavior IV*.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization. *Ieee transactions on evolutionary computation*, 1(1), 67–82. doi : 10.1109/4235.585893.

World Health Organization. (2010). *Birth defects*.

World Health Organization. (2017). Preterm birth fact sheet. Repéré à http://www.who.int/mediacentre/factsheets/fs363/en/.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. & Dean, J. (2016). Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation. *arxiv preprint arxiv :1609.08144*, 1–23. doi : abs/1609.08144.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X. A., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V. & Woodland, P. (2013). The HTK Book. *Journal of chemical information and modeling*, 53(9), 1689–1699. doi : 10.1017/CBO9781107415324.004.

Zabidi, A., Khuan, L. Y., Mansor, W., Yassin, I. M. & Sahak, R. (2010). Classification of infant cries with asphyxia using multilayer perceptron neural network. *2010 2nd international conference on computer engineering and applications, iccea 2010*, 1, 204–208. doi : 10.1109/ICCEA.2010.47.

Zabidi, A., Mansor, W., Khuan, L. Y., Yassin, I. M. & Sahak, R. (2011). Binary particle swarm optimization and f-ratio for selection of features in the recognition of asphyxiated infant cry. *Ifmbe proceedings*, 37, 61–65. doi : 10.1007/978-3-642-23508-5{_}18.

Zhang, Z. (2016). Mechanics of human voice production and control. *The journal of the acoustical society of america*, 140(4), 2614–2635. doi : 10.1121/1.4964509.