

THÈSE

en vue de l'obtention du grade de

Docteur en Informatique

Spécialités : Intelligence Artificielle & Bioacoustique

Les Primitive Neural Networks (PrNN) : un paradigme biomimétique pour l'IA embarquée à ultra basse consommation

Thèse financée par le projet Sylvania ANR-21-CE04-0019,
présentée et soutenue publiquement le 12 décembre 2025, pour l'obtention du

Doctorat de l'Université de Toulon

École Doctorale 548 - Laboratoires IM2NP & LIS
Centre International d'IA en Acoustique Naturelle (CIAN)

par

Adrien DEVERIN

Composition du jury :

Mme.	Patricia DESGREYS	<i>Telecom Paris</i>	Rapporteuse
M.	Olivier BERDER	<i>Université de Rennes</i>	Rapporteur
Mme.	Edith KUSSENER	<i>ISBA TP</i>	Examinatrice
M.	Thierry SORIANO	<i>COSMER & CIAN</i>	Examineur
M.	Arnaud ELGER	<i>AGR Toulouse</i>	Examineur
M.	Valentin GIES	<i>IM2NP & CIAN</i>	Directeur
M.	Hervé GLOTIN	<i>LIS & CIAN</i>	Co-Directeur
M.	Sébastien MARZETTI	<i>IM2NP & CIAN</i>	Encadrant

Remerciements

Je tiens à remercier vivement toutes les personnes que j'ai côtoyées au sein de l'IM2NP, du LIS, et plus largement, à l'Université de Toulon. Merci également à l'Agence Nationale de la Recherche (ANR) SYLVANIA ANR-21-CE04-0019 d'avoir financé ce travail, ainsi qu'à l'ANR-20-CHIA-0014 et au projet TERRA-FORMA .

Ce projet aura représenté trois années de ma vie : trois années de travail, de doutes, d'échecs et de réussites. En bref, trois années de science.

Il est évident qu'une telle thèse ne peut réussir que grâce à un encadrement de qualité. C'est pourquoi je souhaite remercier mon directeur de thèse, Valentin GIES, pour la confiance qu'il m'a accordée, et pour tous ses conseils qui m'ont aidé à en être là aujourd'hui. Merci pour toutes ces discussions scientifiques, ces libertés et cette humanité qui va bien au-delà du cadre professionnel. Je tiens également à remercier mon co-directeur, Hervé Glotin, pour le partage de son expérience et de son temps à de nombreux moments clés de rédaction, ainsi que mon co-encadrant Sébastien Marzetti, fraîchement diplômé avant moi et qui a su me transmettre dès le début ses connaissances. Une pensée particulière va également à Valentin Barchasz, qui a joué pour moi le rôle d'encadrant de l'ombre en m'expliquant un nombre incalculable de choses, ce qui m'a permis de monter en compétences bien plus rapidement. Enfin, merci à Stéphane Chavin et Thierry Lengagne pour leurs discussions passionnantes sur les oiseaux, ainsi qu'à l'ensemble du Centre International d'IA en Acoustique Naturelle (CIAN) pour leur bienveillance.

Je tiens également à remercier mes proches, à commencer par mes parents. Si j'en suis arrivé là, c'est aussi grâce à la chance d'éducation qu'ils m'ont offerte, à leur soutien, et à leur amour pour la personne que j'ai choisi d'être. Merci également à mes amis, qui se reconnaîtront, d'avoir été présents dans ma vie à Toulon, comme d'ailleurs. Cette période de ma vie n'a pas été seulement scientifique : elle a aussi été faite de rencontres humaines, et c'est grâce à vous.

En parlant de rencontres, merci à toi, Rebecca, d'avoir partagé ma vie durant la seconde moitié de ce parcours. Même si tout ne se déroule pas toujours comme on le souhaiterait, je te souhaite beaucoup de bonheur. Merci également pour ton travail de relecture, qui m'a évité de passer pour un illettré auprès de nos chers rapporteurs.

À ce propos, un immense merci à Patricia DESGREYS et Olivier BERDER de m'avoir fait l'honneur d'être les rapporteurs de cette thèse.

Enfin, j'aimerais avoir quelques mots au sujet d'une aventure importante qui s'est déroulée en parallèle de ces trois années : ma participation au sein de l'équipe de France de robotique. Ce fut une expérience inoubliable et j'espère que nous resterons en contact.

Avant de clôturer, je tenais à adresser un merci très spécial à Charline Gauthey, dont le talent a donné vie aux superbes visuels qui embellissent ce manuscrit et que vous découvrirez au fil des pages.

Table des matières

Introduction	2
Partie I - État de l'art	5
1 L'intelligence artificielle en acoustique	6
1.1 Qu'est-ce qu'un réseau de neurones ?	6
1.2 Les méthodes de traitement acoustique	10
1.2.1 Méthodologie de base	10
1.2.2 Les réseaux de neurones de référence	12
1.2.3 Comment optimiser les traitements acoustiques ?	15
1.2.4 Ce qu'il faut en retenir	17
2 Comment réduire la consommation énergétique de l'IA ?	18
2.1 Contexte historique	18
2.2 Les méthodes de décimation neuronale	19
2.2.1 L'élagage	19
2.2.2 La distillation des connaissances	19
2.2.3 Les encodeurs	20
2.2.4 La quantification	20
2.3 L'optimisation matérielle	20
2.4 Les nouvelles couches neuronales	21
2.5 Les architectures neuromorphiques	22
2.6 Résultats concrets	23
2.7 Vers la création des PrNNs	24
3 Le fonctionnement biologique en acoustique	25
3.1 L'oreille externe	26
3.2 L'oreille moyenne	27
3.3 L'oreille interne : le système vestibulaire	29
3.4 L'oreille interne : La cochlée	32
3.5 Liaisons neuronales oreille-cerveau	41

Partie II - Contributions	47
4 Contribution 1 : Une nouvelle métrique d'évaluation - le PPI	48
4.1 Motivations	48
4.2 Formulation mathématique	50
5 Contribution 2 : Architecture des Primitive Neural Networks	54
5.1 Les limites du paradigme actuel	54
5.2 Une analogie hardware des processus biologiques	55
5.3 Une IA embarquable	61
5.4 Une structure permettant l'ultra basse consommation	63
5.4.1 Énergie dépensée pour la partie analogique	63
5.4.2 Comparatif visuel avec l'approche numérique	65
5.4.3 Comparatif énergétique avec l'approche numérique	66
5.4.4 Énergie dépensée pour la partie numérique	67
5.4.5 Bilan	68
6 Contribution 3 : Construire, simuler et optimiser des PrNNs	70
6.1 Objectifs	70
6.2 Structure du code	70
6.3 Interface graphique	73
6.4 Simulation des filtres	77
6.4.1 Étude des filtres BP, LP et HP	77
6.4.2 Implémentation Code	81
6.5 Phases d'apprentissage	84
7 Contribution 4 : Algorithmes d'apprentissage des PrNNs	86
7.1 Apprentissage du bloc d'extraction des primitives (PEx)	86
7.1.1 Tentative d'optimisation à support libre	86
7.1.2 Choix d'apprentissage par élagage d'une structure contrainte	87
7.1.3 Concept de la couche Line Gain	88
7.1.4 Considération sur l'absorption des bandes supprimées	89
7.1.5 Autres exemples d'application	90
7.2 Apprentissage du Trigger	92
7.2.1 Idée générale	92
7.2.2 Modèles envisagés	92

Partie III - Résultats & Perspectives	95
8 Datasets utilisés	96
8.1 Dataset humain	96
8.2 Dataset d'oiseaux	96
8.2.1 Création du jeu de données	97
8.2.2 Pertinence de la méthodologie	100
8.3 Dataset de chauves-souris	102
9 Résultat de l'apprentissage de l'extraction des primitives	103
9.1 Cas d'application 1 : Keyword Spotting (KWS)	103
9.1.1 Choix de l'ordre des filtres passe-bande	103
9.1.2 Évolution de la précision en fonction du nombre de filtre	103
9.1.3 Lien biologique	105
9.1.4 Impact de la répartition initiale	106
9.2 Cas d'application 2 : Reconnaissance d'oiseaux	107
9.2.1 Comparatif des performances entre différents modèles	107
9.2.2 Evolution de l'élagage	108
9.2.3 Analyse biologique	109
9.3 Cas d'application 3 : Présence de chauves-souris	110
10 Résultat de l'apprentissage du Trigger	112
10.1 Pertinence des résultats	112
10.2 Performance des différentes méthodes envisagées	112
11 Efficacité globale des PrNNs	114
11.1 Comparatif global	114
11.2 Conclusion de l'analyse	117
12 Perspectives	118
12.1 Sensibilité variationnelle par attention dynamique	118
12.2 Implémentation matérielle	119
12.3 Vers une intégration sur silicium	119
Conclusion	120
Bibliographie	122

Abstract :

Artificial intelligence (AI) has made remarkable progress in recent years, often surpassing human performance on specific tasks. Yet a critical gap remains : energy consumption. Comparing the operational cost of artificial neural networks with the synaptic cost of the biological brain reveals that human intelligence is, on average, $10\,000\times$ more energy-efficient than its artificial counterpart.

Beyond the environmental implications, this disparity is a major obstacle to fully autonomous systems. This thesis therefore aims to design a new AI architecture for passive recognition under ultra-low-power constraints. We focus on acoustic recognition, with an application to automated bird monitoring in forests to estimate the size and spatial distribution of target populations across defined geographic areas.

We begin by analyzing in detail the biological mechanisms underlying sound recognition and contrasting them with conventional AI architectures. Based on this analysis, we propose a new architecture : the *Primitive Neural Networks* (PrNN), a patented approach [1] that leverages low-power biological processes through analog primitives and a dynamic noise-adaptation mechanism. PrNN is inspired by the mammalian cochlea, and its learning draws on biological principles of co-evolution between sensory organs and neural circuits.

We further introduce an evaluation framework that assesses system performance jointly by accuracy and energy consumption. The resulting metric, the *Performance–Power Index* (PPI), serves as the optimization objective at the algorithmic level.

Experimental results demonstrate that PrNNs achieve a three-order-of-magnitude reduction in energy consumption compared to conventional approaches, even in the absence of an application-specific integrated circuit (ASIC). These results open the door to a new generation of intelligent systems capable of operating autonomously and sustainably.

Beyond the applications presented here, this work illustrates how biological inspiration can help reimagine the foundations of artificial intelligence, bringing machine performance closer to human-level efficiency.

Keywords : Embedded AI ; Biomimicry ; Ultra-low-power ; Audio signal processing ; Passive acoustic recognition.

Résumé :

Le domaine de l'intelligence artificielle (IA) a connu des avancées spectaculaires ces dernières années, allant jusqu'à surpasser l'humain dans certaines tâches. Toutefois, une différence majeure demeure : la consommation énergétique. En comparant le coût opérationnel des réseaux de neurones artificiels au coût synaptique du cerveau biologique, on constate que l'intelligence humaine est en moyenne 10 000 fois plus efficace que son équivalent artificiel.

Au-delà des implications environnementales évidentes que soulève cette disparité, celle-ci constitue également un frein majeur au développement de systèmes autonomes. Dans ce contexte, l'objectif scientifique de cette thèse est de concevoir une nouvelle architecture d'IA permettant des tâches de reconnaissance passive sous la contrainte d'ultra-basse consommation énergétique. Plus particulièrement, nous nous concentrons sur la problématique de la reconnaissance acoustique, avec pour cas d'application la surveillance automatique d'oiseaux en milieu forestier, en vue d'estimer la taille et la répartition de certaines populations au sein d'une zone géographique donnée.

Dans une première étude, nous analysons en détail les mécanismes biologiques impliqués dans la reconnaissance sonore et les mettons en perspective avec les architectures classiques d'IA. Cette analyse nous conduit à la proposition d'une nouvelle architecture : les *Primitive Neural Networks* (PrNNs), une invention brevetée [1] qui tire avantage des processus faiblement énergivores du monde biologique en utilisant des primitives analogiques et un mécanisme d'adaptation dynamique au bruit. Son fonctionnement est bio-inspiré par la cochlée des mammifères et son mécanisme d'apprentissage repose principalement sur des principes biologiques de co-évolution entre les organes sensoriels et les structures neuronales.

Cette contribution s'accompagne de l'introduction d'un nouveau cadre d'évaluation dans lequel la performance d'un système ne se mesure plus uniquement en termes de précision, mais également en fonction de l'énergie consommée. Cette métrique est formalisée sous l'appellation de *Performance-Power Index* (PPI) et sert d'objectif d'optimisation sur le plan algorithmique.

Les expériences menées démontrent que les PrNNs atteignent une efficacité similaire à celle des approches classiques, tout en réduisant la consommation énergétique d'un facteur 1 000, et ce, sans recours à une implémentation ASIC (*application-specific integrated circuit*). Ces résultats ouvrent la voie à une nouvelle génération de systèmes intelligents capables d'opérer de manière autonome et durable.

Au-delà des applications présentées, ce travail illustre comment l'inspiration biologique peut contribuer à réinventer les fondations de l'intelligence artificielle, en rapprochant ses performances de l'efficacité de l'être humain.

Mots-clés : Intelligence artificielle embarquée ; Biomimétisme ; Ultra-basse consommation ; Traitement du signal audio ; Reconnaissance acoustique passive.

Liste des abréviations

ADC	Analog-to-Digital Converter, assure l'échantillonnage du signal analogique.
AI	Artificial Intelligence (voir la section : Avant propos).
AOP	Amplificateur Opérationnel, composant analogique fondamental.
ASIC	Application-Specific Integrated Circuit, circuit intégré dédié à une application spécifique, souvent conçu sans microcontrôleur afin de maximiser l'efficacité énergétique.
AUC-PR	(ou PR-AUC) Area Under the Precision–Recall Curve, métrique.
BP	Band-Pass (Filter).
CCE	Cellules Ciliées Externes, cellules de la cochlée contribuant à l'amplification mécanique du signal auditif.
CCI	Cellules Ciliées Internes, cellules de la cochlée contribuant au filtrage fréquentiel du signal auditif.
CNN	Convolutional Neural Networks.
CRNN	Convolutional Recurrent Neural Networks, architecture combinant couches convolutionnelles et récurrentes.
FFT	Fast Fourier Transform (similaire à STFT), algorithme largement utilisé en analyse spectrale et permettant la décomposition temps-fréquences.
GRU	Gated Recurrent Unit, couche récurrente avec une complexité réduite par rapport aux LSTM.
KWS	Keyword Spotting, tâche consistant à détecter la présence de mots-clés prédéfinis dans un signal audio continu.
LPPD	Low-Pass Peak Detector.
MAC	Multiply–Accumulate Operation, opération fondamentale des réseaux de neurones ; souvent comptabilisée en MMAC (millions de MAC).
MCU	Microcontroller Unit, système embarqué intégrant processeur, mémoire et périphériques sur une même puce.
PCEN	Per-Channel Energy Normalization, technique de normalisation spectrale.
PE_x	Primitives Extraction, étape consistant à extraire des caractéristiques élémentaires à partir de signaux bruts.
PPI	Performance–Power Index, indicateur quantifiant le compromis entre performance algorithmique et consommation énergétique.
PrNN	Primitive Neural Networks, sujet du manuscrit, voir Contribution 2.
RNN	Recurrent Neural Networks.
SNN	Spiking Neural Networks, réseaux de neurones bio-inspirés utilisant des impulsions discrètes (spikes) pour le calcul.
SNR	Signal-to-Noise Ratio, exprimé généralement en décibels (dB).
SoC	System-on-Chip, circuit intégrant sur une même puce l'ensemble des composants nécessaires à un système électronique complet.
ULP	Ultra-Low Power, consommation énergétique de l'ordre de 100 μ W.

Avant propos

Puisqu'il s'agira d'un thème central dans ce manuscrit, il convient d'apporter au préalable une réponse, à minima contextuelle, à la question : *Qu'est-ce qu'une IA ?*

Bien que ce terme soit largement diffusée dans la littérature comme dans le langage courant, l'appellation « intelligence artificielle » (ou IA) recouvre en réalité des acceptions très diverses, allant de la simple *machine à états* aux *Large Language Models* (LLM). Il ne fait nul doute que cette délimitation théorique pourrait être longuement débattue, mais il ne s'agit pas de l'objectif de cette thèse.

Ainsi, nous utiliserons le terme IA dans un sens restrictif, pour désigner l'ensemble des architectures incorporant un réseau de neurones pour la tâche de classification (voir Section 1.1 : *Qu'est ce qu'un réseau de neurone ?*).

Par ailleurs, sauf indication contraire, l'expression « consommation d'une IA » fera référence à la consommation énergétique totale requise pour son fonctionnement, englobant à la fois celle des composants matériels nécessaires à l'acquisition et au pré-traitement des données, ainsi que celle mobilisée par les traitements effectués au sein de ses réseaux neuronaux.

*Le meilleur moyen de prédire
l'avenir, c'est de l'inventer.*

Alan Kay

Introduction

Contexte de la thèse

L'utilisation de l'intelligence artificielle a connu une croissance impressionnante au cours des dernières années, au point d'être aujourd'hui au cœur de notre société. Cette progression a été rendue possible grâce au développement continu de nouvelles architectures neuronales (CNN, RNN, Transformers, modèles multimodaux), à la disponibilité croissante de jeux de données massifs, ainsi qu'à l'augmentation des capacités de calcul.

Parmi les nombreuses thématiques explorées, celle de la reconnaissance acoustique occupe une place particulièrement importante. Elle recouvre un large spectre d'applications, pouvant aller de la détection d'anomalies sonores en contexte industriel (identification de fuites ou de défaillances) jusqu'à la transcription automatique du langage, ouvrant la voie à des interactions naturelles homme-machine (le désormais emblématique « Ok Google »).

C'est dans ce cadre que s'inscrit la thèse présentée, financée par l'ANR SylvanIA et menée en partenariat avec les laboratoires LIS et IM2NP. Plus spécifiquement, nous nous intéressons au champ de la bioacoustique, avec pour objectif une détection automatique d'espèces ciblées d'oiseaux. L'enjeu principal est de surveiller la dynamique de leurs populations et de mieux comprendre certains phénomènes migratoires.

Une problématique mondiale

L'un des événements fondateurs de l'ère de l'intelligence artificielle fut sans doute la victoire aux échecs de l'IA *Deep Blue* face à Garry Kasparov, alors multiple champion du monde. Bien que la performance fût impressionnante, le duel restait cependant inégal. En effet, l'énergie requise au fonctionnement de *Deep Blue* était plus de mille fois celle utilisée par un cerveau humain (près de 30 kW contre seulement 20 W). De surcroît, il convient de rappeler qu'une partie significative de cette énergie cérébrale est dédiée aux fonctions vitales, si bien que la proportion effectivement mobilisée pour les capacités cognitives reste encore inférieure. Autrement dit, la machine a eu besoin de l'équivalent énergétique de plusieurs milliers d'hommes pour en vaincre un seul.

Même si les technologies ont considérablement évolué depuis, ce déséquilibre demeure l'une des problématiques majeures du domaine : nos approches traditionnelles restent beaucoup

trop énergivores en comparaison de l'efficacité biologique. Aujourd'hui, la plupart des spécialistes estiment que nous avons atteint un véritable « plafond de verre », qu'il ne sera possible de dépasser qu'avec un changement profond de nos architectures d'IA.

Franchir cette limitation représente un double enjeu. D'une part, un enjeu écologique : l'intelligence artificielle est désormais omniprésente dans notre société et sa consommation énergétique croissante s'avère difficilement soutenable à long terme. D'autre part, un enjeu d'usage : une telle consommation rend impossible le déploiement d'algorithmes sur des dispositifs fonctionnant sur batterie pendant de longues périodes.

Pour ce manuscrit, ce second point est particulièrement crucial, car il constitue un obstacle direct à notre application de recherche, qui prévoit de déployer des capteurs autonomes dans des zones dépourvues d'alimentation électrique continue.

Objectif de la thèse

Cette thèse a pour objectif de concevoir une architecture d'intelligence artificielle capable de traiter des données acoustiques en temps réel tout en maintenant une consommation énergétique extrêmement faible (inférieure à 250 μ W).

Le modèle proposé, nommé Primitive Neural Network (PrNN), adopte une structure biomimétique reposant sur des traitements analogiques en amont de la conversion numérique. Cette approche permet d'extraire des primitives du signal inaccessibles par de simples produits scalaires, tout en évitant des inférences numériques inutiles lors d'événements rares, réduisant ainsi la consommation moyenne.

Les contributions de cette thèse s'inscrivent dans le développement d'un paradigme d'apprentissage mixte, agissant à la fois sur la partie analogique et sur le réseau neuronal numérique. Nous visons une consommation énergétique de 100 à 1000 fois inférieure aux approches traditionnelles.

Partie I

État de l'art

L'intelligence artificielle en acoustique

Toute technologie suffisamment avancée est indiscernable de la magie.

Arthur C. Clarke

1.1 Qu'est-ce qu'un réseau de neurones ?

Structure

Les réseaux de neurones artificiels sont des modèles computationnels dont la conception fût originellement inspirée du fonctionnement des neurones biologiques, avant de se rapprocher des méthodes statistiques [7]. Ici, les neurones biologiques sont remplacés par des *perceptrons* : des entités mathématiques qui effectuent une combinaison linéaire de leurs entrées.

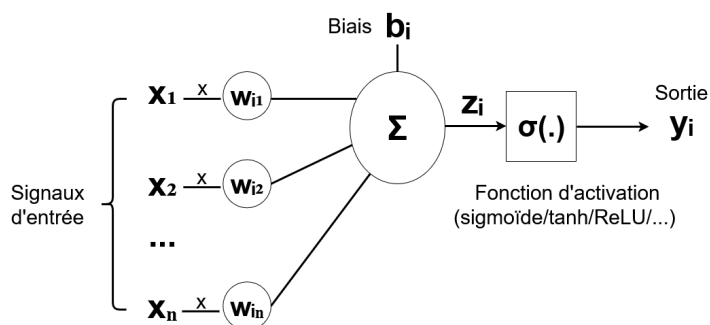


FIGURE 1.1 – Schéma du fonctionnement d'un perceptron dans un réseau de neurones artificiel.

Pour un perceptron donné, la sortie z_i s'écrit :

$$z_i = \sum_{j=1}^n w_{ij} x_j + b_i \quad , \quad (1.1)$$

où x_j désigne les entrées, w_{ij} les poids associés, et b_i le biais. Ce signal est ensuite transformé par une fonction d'activation non linéaire $\sigma(\cdot)$ pour produire la sortie y_i .

Les perceptrons sont organisés en couches qui composent le réseau. Lors d'une *inférence*, la donnée d'entrée (image, signal acoustique, mesure numérique, etc.) est représentée sous la forme d'un tenseur et propagée de couche en couche jusqu'à produire un vecteur de sortie. Dans un problème de classification, ce vecteur a généralement une dimension égale au nombre de classes et ses valeurs sont normalisées (entre zéro et un) pour être interprétées comme des probabilités d'appartenance aux différentes classes.

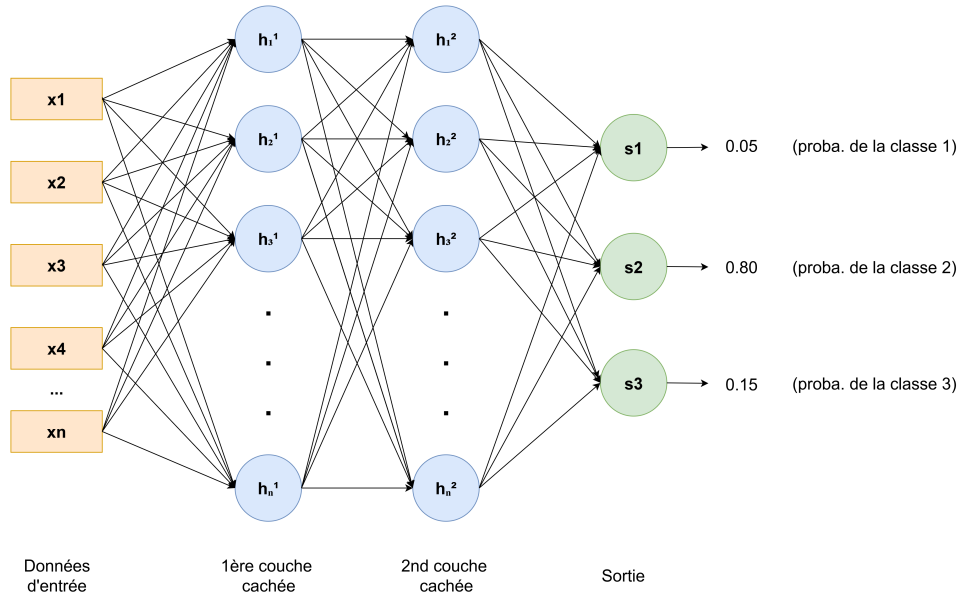


FIGURE 1.2 – Schéma simplifié d'un réseau de neurones artificiel à plusieurs couches (MLP).

Apprentissage

L'apprentissage d'un réseau de neurones repose sur un mécanisme d'optimisation appelé la *rétropropagation* [8]. À chaque itération, la sortie du réseau $\hat{y}$ est comparée à la sortie attendue y au moyen d'une fonction de perte $\mathcal{L}(y, \hat{y})$. L'erreur obtenue est alors propagée dans le sens inverse de l'inférence grâce à une formule mathématique que l'on appelle « la règle de la chaîne ». Grâce à cette formule, on obtient en tout point du réseau un gradient, noté $\frac{\partial \mathcal{L}}{\partial w_{ij}}$ ou $\frac{\partial \mathcal{L}}{\partial b_i}$, qui correspond à une direction vers laquelle doit évoluer le poids (resp. le biais) associé pour que l'erreur finale diminue. On qualifie alors par le terme « descente de gradient » le fait de modifier l'ensemble des poids de notre réseau le long de ces vecteurs :

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial \mathcal{L}}{\partial w_{ij}}, \quad b_i \leftarrow b_i - \eta \frac{\partial \mathcal{L}}{\partial b_i}, \quad (1.2)$$

où η est le taux d'apprentissage. Ce processus, répété sur de larges ensembles de données annotées, permet au réseau d'améliorer progressivement ses performances jusqu'à convergence.

Aujourd'hui, la plupart des systèmes d'intelligence artificielle reposent sur des réseaux de neurones dans leur chaîne perceptive. Il existe plusieurs types de couches, plus ou moins adaptées à différents type de tâches. Par exemple, les couches illustrées précédemment, appelées couches denses, conviennent généralement aux données tabulaires ou pour des combinaisons finales d'informations. Les couches convolutionnelles (CNN) sont quand à elles spécialisées dans l'extraction de motifs locaux et ont révolutionné la vision par ordinateur. Les couches récurrentes (RNN, LSTM, GRU) sont adaptées au traitement des données séquentielles telles que le langage naturel ou les signaux acoustiques. Enfin, les architectures à base de transformeurs (LLM) se sont imposées comme une avancée récente majeure pour la modélisation de séquences longues et complexes, avec des applications marquantes en traduction automatique, en compréhension de documents ou en reconnaissance vocale.

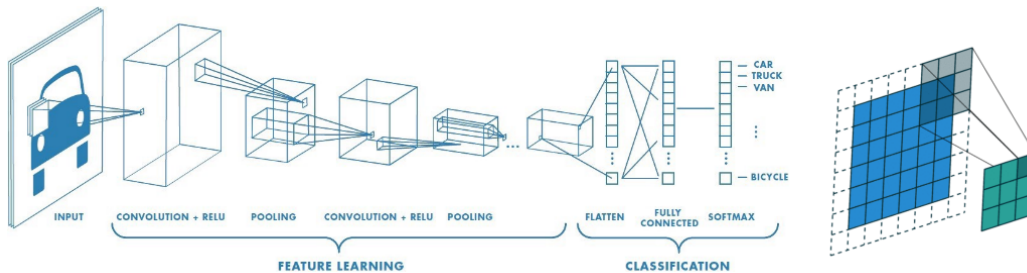


FIGURE 1.3 – Exemple de réseau convolutionnel (CNN) appliqué à une image.

Évaluation d'un réseau de neurones

L'évaluation d'un réseau de neurones constitue une étape essentielle : elle permet de déterminer le moment où l'apprentissage peut être arrêté, de vérifier si le modèle répond aux objectifs fixés et de comparer ses performances avec celles de d'autres approches. Elle repose généralement sur l'utilisation de jeux de données distincts de ceux employés pour l'entraînement (validation et test), afin de garantir une mesure objective de la capacité de généralisation du modèle.

Les métriques de classification s'appuient sur quatre catégories de prédictions :

- **Les TP (True Positives)** : cas positifs correctement identifiés.
- **Les FP (False Positives)** : cas négatifs incorrectement identifiés comme positifs.
- **Les TN (True Negatives)** : cas négatifs correctement identifiés.
- **Les FN (False Negatives)** : cas positifs incorrectement identifiés comme négatifs.

La métrique la plus courante est l'**Accuracy**. Elle est définie comme la proportion de prédictions correctes sur l'ensemble des exemples :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (1.3)$$

Dans notre contexte, cette métrique peut s'avérer trompeuse car il existe généralement un fort déséquilibre entre les classes (les moments de silence dominant largement les événements d'intérêt). Par exemple, si 29/30 des événements sont négatifs, un modèle naïf qui prédirait systématiquement l'absence de classe atteindrait une accuracy de 96,7% sans jamais apprendre à détecter les cas positifs.

Afin de ne pas biaiser l'interprétation de nos résultats, nous utiliserons pour ces jeux de données d'autres métriques plus représentatives :

- La **Précision**, qui mesure la proportion de prédictions positives correctes :

$$\text{Précision} = \frac{TP}{TP + FP}. \quad (1.4)$$

- Le **Rappel**, qui mesure la proportion de cas positifs correctement identifiés :

$$\text{Rappel} = \frac{TP}{TP + FN}. \quad (1.5)$$

- Le **F1-score**, moyenne harmonique entre précision et rappel, qui évalue le compromis entre ces deux aspects :

$$\text{F1-score} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}. \quad (1.6)$$

En pratique, la décision de classer une observation comme positive repose sur un *seuil d'acceptation*, compris entre 0 et 1. Lorsque l'une des sortie du réseau dépasse ce seuil, la classe qui y est associée est jugée positive. Faire varier ce seuil influence donc directement la précision et le rappel : un seuil très bas maximise le rappel au détriment de la précision, et inversement en cas de seuil élevé. Pour cette raison, il est fréquent de tracer la **courbe Précision–Rappel** en fonction du seuil et d'utiliser l'aire sous cette courbe (**AUC–PR**, *Area Under Curve – Precision–Recall*) comme mesure objective et robuste. C'est ce que nous ferons au sein de ce manuscrit.

D'autres métriques existent et peuvent s'avérer plus pertinentes selon les tâches (comme le *mAP@50-95* en détection d'objets visuels ou le *Mixture of Experts* pour les modèles génératifs de texte), mais elles sont hors sujet ici.

Dans notre contexte, il est en revanche indispensable de compléter l'évaluation prédictive par une évaluation liés à l'**efficacité opérationnelle du modèle**. Deux dimensions

sont particulièrement critiques : (1) la capacité à fonctionner en **temps réel**, condition nécessaire pour la détection d'événements acoustiques continus, et (2) la **consommation énergétique**, qui peut être évaluée soit à partir du nombre d'opérations élémentaires effectuées par le réseau, exprimées en **MAC** (*Multiply-Accumulate Operations*, constituant fondamental des produits scalaires), soit directement en puissance dissipée, mesurée en watts ($J.s^{-1}$).

Ces considérations sont fondamentales ; c'est pourquoi l'une des premières contributions de cette thèse consistera à proposer une nouvelle métrique d'évaluation intégrant à la fois la performance de classification et la consommation énergétique d'un modèle.

1.2 Les méthodes de traitement acoustique

1.2.1 Méthodologie de base

Réaliser une tâche de reconnaissance acoustique revient à transformer la sortie analogique d'un microphone (*le signal audio*) en une réponse (souvent numérique et normalisée) correspondant à la certitude de présence d'une classe. En pratique, on scinde les réponses sur des fenêtres temporelles du signal (de 1 à 5 secondes) afin d'avoir un contexte suffisant pour émettre de telles affirmations. L'acquisition de ces fenêtres temporelles se fait à l'aide d'un ADC (*analog digital converter*) qui va échantillonner à haute fréquence (typiquement 16 kHz pour la voie humaine) la sortie du microphone. Cet ensemble de points est ensuite traité par le reste de l'IA.

Néanmoins, l'application directe d'un réseau de neurones sur ces échantillons s'avère souvent inefficace. La raison est avant tout physique : c'est l'évolution des composantes énergétiques fréquentielles qui permet de distinguer les sons [9] et non l'amplitude temporelle. Or, mathématiquement, l'énergie spectrale ne peut pas être obtenue par de simples combinaisons linéaires sur l'ensemble des points fournis, ce qui rend difficile une reconnaissance acoustique performante par le réseau de neurones. Ce principe fondamental est la raison pour laquelle la quasi-totalité des approches en IA acoustique appliquent un prétraitement du son, destiné à extraire ces caractéristiques spectrales, avant de faire leur analyse par un réseau de neurones.

Pour cette extraction, on utilise presque systématiquement une **Transformée de Fourier Rapide** glissante (*ou short-time Fourier transform, STFT*) pour obtenir un spectrogramme, auquel on applique d'autres transformations, destinées à améliorer la robustesse et la compacité de la représentation.

D'autres transformations plus marginales, telles que la transformée en ondelettes ou la distribution de Wigner-Ville, permettent une analyse temps-fréquence plus fine, mais restent plus complexes à mettre en œuvre. La distribution de Wigner-Ville présente notamment des interférences croisées, tandis que les approches par ondelettes nécessitent un ajustement précis des fonctions de base et des échelles, souvent spécifique à la nature du signal étudié. En outre, leur coût computationnel est très élevé, ce qui limite fortement leur usage, notamment dans le cadre d'applications embarquées en traitement temps réel.

Parmi les méthodes les plus répandues, on distingue notamment (Fig. 1.4) :

- le **Mel-Spectrogramme** qui est obtenue par projection du spectrogramme sur une échelle Mel, plus adaptée à la perception auditive humaine ;
- le **Log Filter Bank Energies-Delta (LFBE-delta)**, qui ajoute aux spectres les dérivées première et seconde afin de plus facilement capturer les variations temporelles ;
- la **Per-Channel Energy Normalization (PCEN)**, une alternative à la compression logarithmique (dB), qui normalise l'énergie par canaux du spectre afin de décorrélérer les bandes fréquentielles et ainsi améliorer la robustesse aux interférences acoustiques ;
- les **Mel-Frequency Cepstral Coefficients (MFCC)**, qui résultent d'une transformée en cosinus discrète (DCT) sur le spectre Mel afin de réduire la dimensionnalité et d'être plus robuste face aux bruits et aux variations d'environnement.

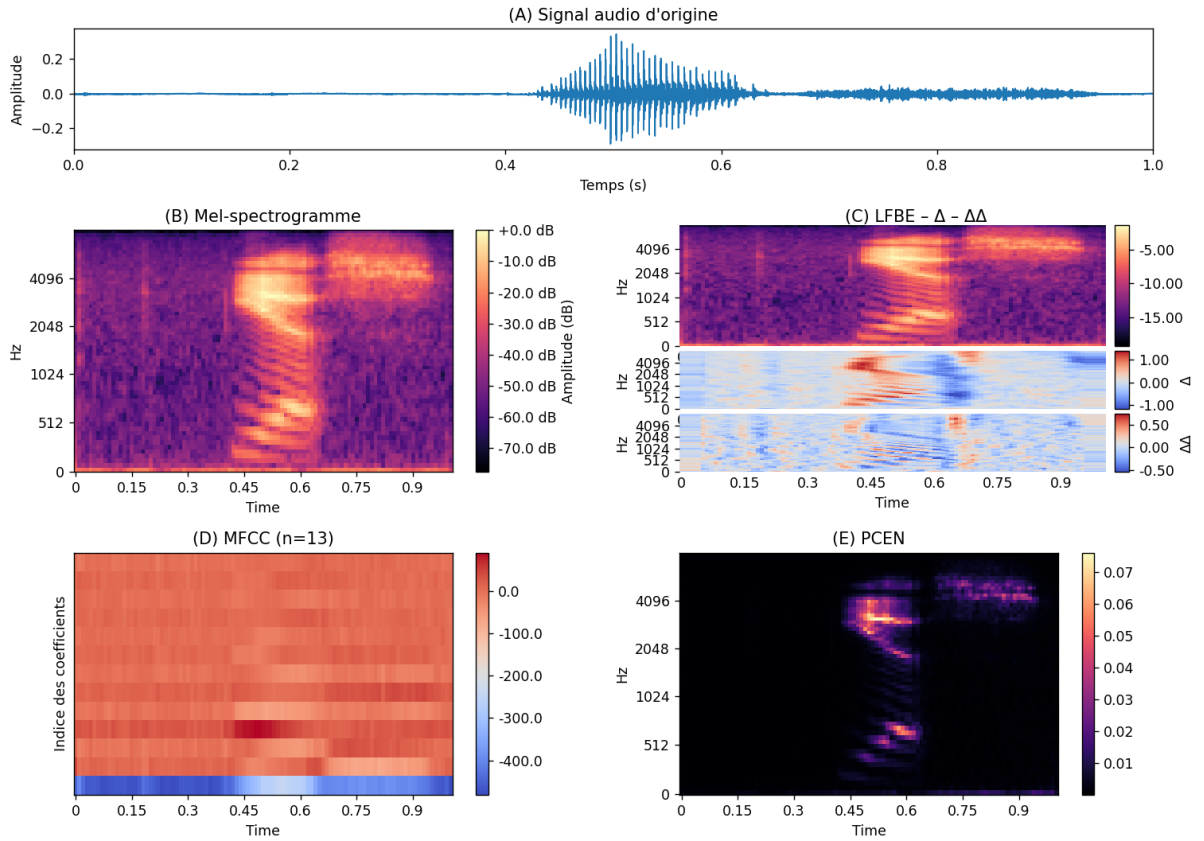


FIGURE 1.4 – Comparaison des principales méthodes de prétraitement acoustique : (a) signal d'origine ; (b) spectrogramme Mel ; (c) LFBE-delta ; (d) MFCC ; (e) PCEN. Le signal dure 1 s, est échantillonné à 16 kHz, et correspond au mot « yes ».

Structuellement, ces prétraitements transforment un vecteur unidimensionnel (le signal audio) en une représentation temps–fréquence bidimensionnelle (analogue à une image). Le traitement du son par un réseau de neurones devient alors similaire à une tâche de classification d'images. Ces architectures combinent souvent des couches convolutionnelles et récurrentes (LSTM ou GRU) afin de modéliser la dynamique temporelle des motifs acoustiques.

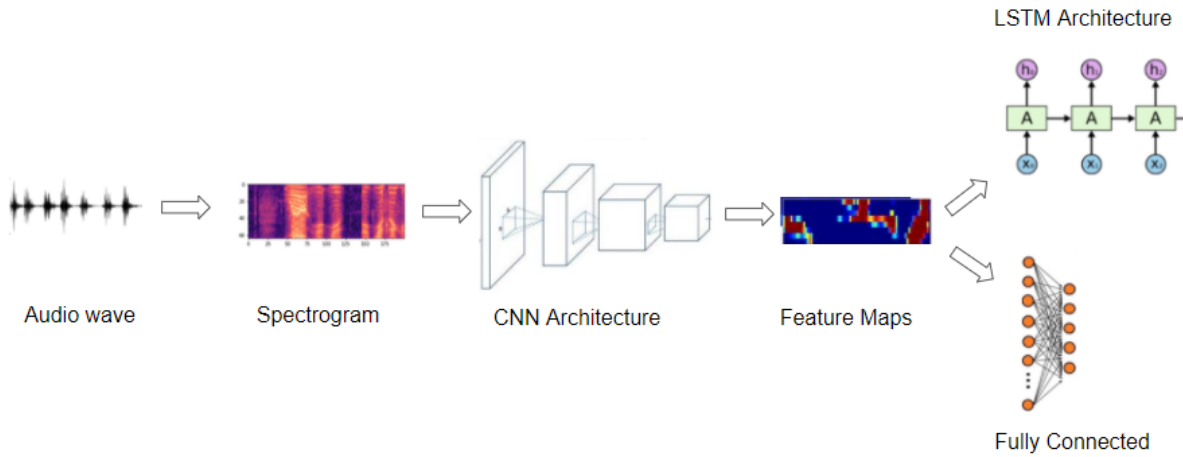


FIGURE 1.5 – Processus typique du traitement acoustique par une intelligence artificielle.

1.2.2 Les réseaux de neurones de référence

Au sein de ce manuscrit, nous allons devoir comparer nos travaux avec le reste de l'état de l'art. Pour ce faire, nous avons sélectionné quelques IAs connues dans le domaine ayant des réseaux de neurones open source. Cette section décrit ces structures et leurs particularités.

BirdNET

BirdNET [10] est l'un des systèmes d'intelligence artificielle les plus cités en bioacoustique pour la reconnaissance d'oiseaux [11–13]. Son architecture s'inspire d'EfficientNet-B1 et opère sur des signaux audio échantillonnés à 48 kHz. À partir de fenêtres d'inférence de 3 s, le pipeline extrait deux mel-spectrogrammes de taille 96×511 sur des bandes fréquentielles complémentaires.

Le modèle peut s'exécuter en ligne ou en embarqué sur des plateformes de type Raspberry Pi. La base d'entraînement utilisée est issue du corpus open-source Xeno-Canto [14] et comporte plus de 3000 espèces d'oiseaux. Dans ce manuscrit, tous les résultats obtenus par ce modèle proviennent d'un transfert d'apprentissage réalisé sur notre jeu de données.

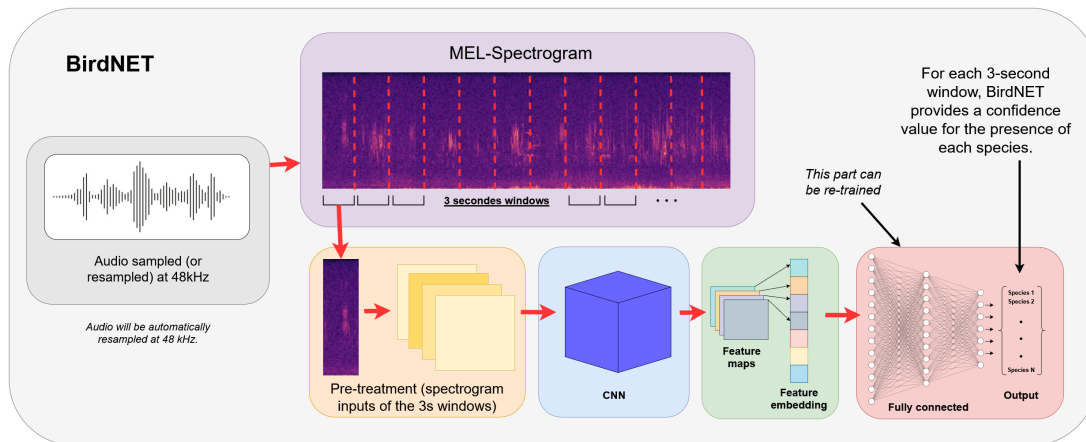


FIGURE 1.6 – Schéma de traitement acoustique par le modèle BirdNet

Google-Perch

Perch¹ [15] est aujourd’hui le modèle ayant obtenu les meilleures performances en reconnaissance acoustique d’oiseaux. Créé par Google, il a été entraîné pour classer près de 15 000 espèces, dont 10 000 d’oiseaux. Comme BirdNET, son architecture est elle aussi inspirée d’EfficientNet-B1 à la différence que l’entrée provient cette fois de signaux audio de 5 secondes, échantillonnés à 32 kHz. Les entrées utilisées sont des PCEN Mel-spectrogram de taille 128×500 .

YAMNet

YAMNet [16] est un modèle de classification audio généraliste, construit sur l’architecture MobileNet-v1. Il a été initialement entraîné sur le jeu de données AudioSet (issus majoritairement de YouTube), couvrant un large éventail de classes sonores, dont certaines liées à la bioacoustique. Le modèle prend en entrée des signaux de 0.96 seconde, échantillonnés à 16 kHz, pour en faire des MEL-Spectrogramme de taille 64×96 .

Bien que conçu pour la classification générale, YAMNet peut être utilisé comme extracteur de caractéristiques. Les représentations extraites (embeddings) constituent alors les entrées d’entraînement pour des couches de classification adaptées à des tâches spécifiques. La raison pour laquelle nous avons sélectionné ce modèle est l’existence de deux variantes quantifiées et allégées pour l’embarqué, disponibles dans le *Model Zoo* de STMicroelectronics². Nous désignerons ces variantes sous les noms de Yamnet-1024 et Yamnet-256 (embedding réduit) dans nos expérimentations.

BenBogart

Ce modèle, développé par Ben Bogart³ [17], s’appuie sur un réseau de neurones convolutifs pour la classification d’espèces d’oiseaux à partir d’enregistrements audio. Il a été initialement entraîné sur le jeu de données du concours Kaggle (*Cornell Birdcall Identification*), comprenant près de 20 000 enregistrements pour 264 espèces. Ce corpus a ensuite été enrichi avec plus de 40 000 enregistrements supplémentaires issus de la base Xeno-Canto.

Initialement, les fichiers audio sont échantillonnés à 22 050 Hz et découpés en segments de 10 secondes. Toutefois, nous utilisons une variante avec des entrées de 5 secondes pour produire des images de taille 256×212 , ce qui allège le coup computationnel et donne des résultats similaires.

SSL-BirdSoundRecognition

Le modèle SSL-BSR⁴ [18], développé par META AI, a été entraîné sur des vocalisations d’oiseaux enregistrées dans les forêts subtropicales de Taïwan. Il s’appuie sur *Audio-*

1. <https://www.kaggle.com/models/google/bird-vocalization-classifier/tensorFlow2/bird-vocalization-classifier>

2. https://github.com/STMicroelectronics/stm32ai-modelzoo/blob/main/audio_event_detection/yamnet

3. https://github.com/benbogart/bird_vocalization_classification

4. <https://github.com/mtshiao/ssl-BirdSoundRecognition>

MAE, un schéma encodeur–décodeur par Transformers préentraîné en auto-supervision par masquage. Les entrées sont des segments audio de 10 s échantillonnés à 16 kHz ; les spectrogrammes correspondants sont découpés en patchs de 16×16 avant d’être fournis au modèle. Le préapprentissage consiste à masquer aléatoirement une fraction des patchs et à reconstituer le spectrogramme, de façon à apprendre des représentations générales réutilisables pour la classification aval.

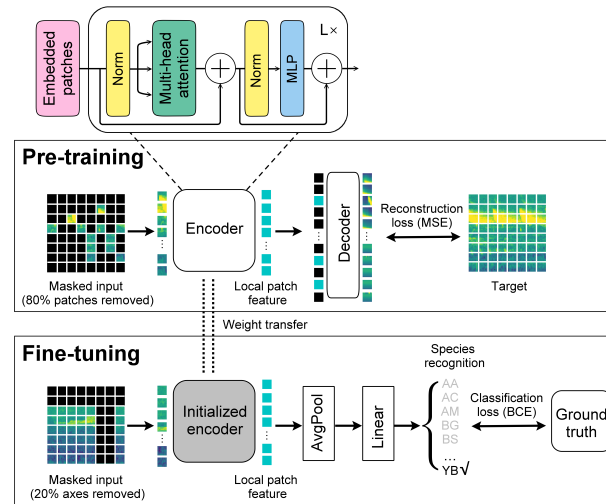


FIGURE 1.7 – Schéma du modèle SSL-BirdSoundRecognition. *Figure issue du repo GitHub*

Whisper

Dans la même dynamique, Whisper (OpenAI) [19] est un système de reconnaissance fondé sur un Transformer encodeur–décodeur de la parole humaine. Il opère sur des segments d’environ 30 secondes sous forme de Mel-spectrogrammes et a été entraîné à grande échelle (plusieurs centaines de milliers d’heures) pour la transcription, l’identification de la langue et la traduction automatique vers l’anglais. Les plus petits modèles s’exécutent en temps quasi réel sur des systèmes de type Raspberry Pi.

Yolo

Puisque le traitement du son peut être assimilé à un traitement d’image, nous avons testé des architectures Yolo (You Only Look Once) [20, 21], des modèles open source, reconnu comme une référence dans le domaine de la reconnaissance visuelle. Nous utilisons comme entrée un MEL-spectrogramme de taille 64×300 , obtenue via une FFT sur des fenêtres de 3 secondes.

L’avantage principal de ce modèle est qu’il ne fournit pas seulement la présence d’une espèce, mais aussi sa position et sa durée dans le spectrogramme. Cependant, afin de ne pas biaiser les résultats avec une tâche plus complexe, nous n’effectuons que la détection de présence.

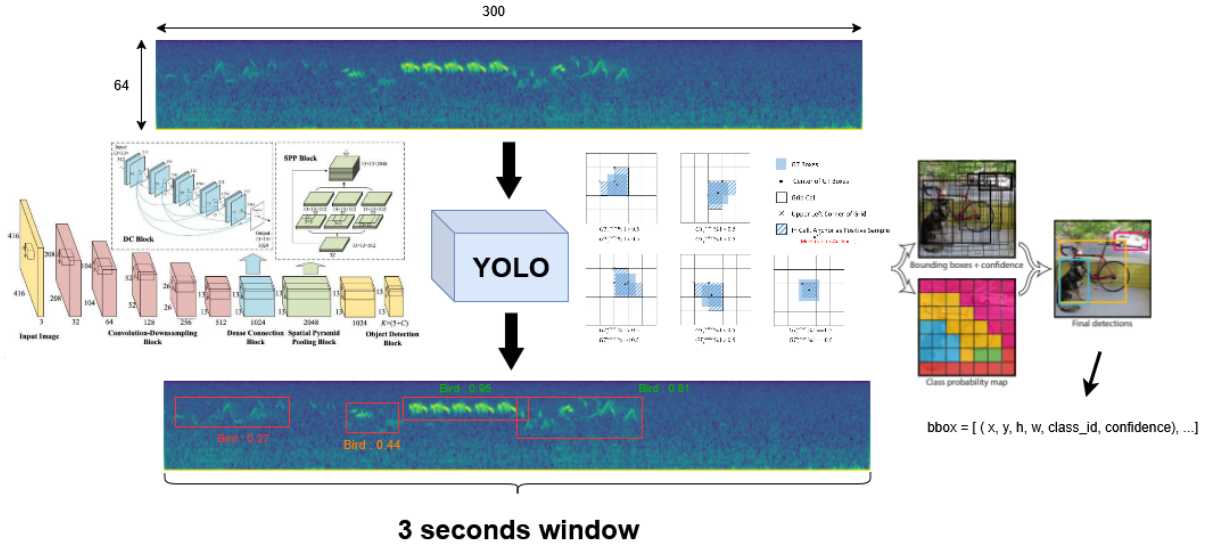


FIGURE 1.8 – Schéma de traitement du modèle YoloV8

1.2.3 Comment optimiser les traitements acoustiques ?

Il convient de noter que les choix de représentation évoqués précédemment sont relativement arbitraires ; l'échelle Mel ayant même été modifiée à plusieurs reprises au cours de son histoire [22, 23]. L'idéal dans nos travaux serait de parvenir à apprendre la représentation sonore optimale, afin d'améliorer la performance finale sur une tâche ciblée. Il existe plusieurs travaux qui ont tenté d'aller dans ce sens. Parmi les premières propositions on retrouve l'idée d'appliquer directement plusieurs grands noyaux convolutifs 1D sur les fenêtres sonores (*time-dilated convolutions*) ; ces filtres étant différentiables, ils peuvent être appris par descente de gradient. Néanmoins, cette approche présente deux limites majeures : (i) elle accroît fortement la complexité paramétrique et computationnelle, induisant un apprentissage plus long, une latence et un surcoût énergétique ; (ii) en raison des contraintes mathématiques précédemment mentionnées, l'extraction des composantes spectrales reste imprécise, ce qui la rend souvent moins performante que les représentations Mel.

L'idée sous-jacente demeure toutefois pertinente, et a été revisitée en 2018 par Mirco Ravanelli et Yoshua Bengio avec la conception de **SincNet** [24]. L'innovation majeure de ces travaux réside dans une modification de la première couche convolutive : au lieu d'apprendre librement tous les coefficients des filtres, SincNet contraint chaque filtre à être défini comme une approximation lisse d'une fonction passe-bande au moyen de la fonction sinc (différentiable). Concrètement, chaque filtre est représenté uniquement par deux paramètres, f_1 et f_2 , correspondant respectivement aux fréquences de coupure basse et haute. Sa réponse impulsionnelle est donnée par :

$$h(t; f_1, f_2) = 2f_2 \operatorname{sinc}(2\pi f_2 t) - 2f_1 \operatorname{sinc}(2\pi f_1 t), \quad \text{avec} \quad \operatorname{sinc}(x) = \frac{\sin(x)}{x}. \quad (1.7)$$

Par rapport à l'approche précédente, cette méthode requiert beaucoup moins de paramètres (seulement deux par filtre) et demeure plus facilement interprétable, puisqu'à chaque filtre correspond une bande de fréquences identifiable. Cependant, bien que l'idée soit théoriquement très prometteuse, les performances pratiques de SincNet sur des tâches complexes restent équivalentes, voire légèrement inférieures, à celles des architectures classiques basées sur spectrogrammes et CNN.

En 2021, dans le prolongement de cette idée, a été proposé par Zeghidour et coll. **LEAF** (*LEarnable Audio Frontend*) [25] : un frontend audio différentiable et entièrement apprenable. Alors que SincNet s'arrêtait à l'apprentissage des filtres passe-bande en guise de première couche, LEAF va plus loin en proposant un pipeline complet, de l'audio brut jusqu'à une représentation temps-fréquence, mais appris de bout en bout (Figure 1.9).

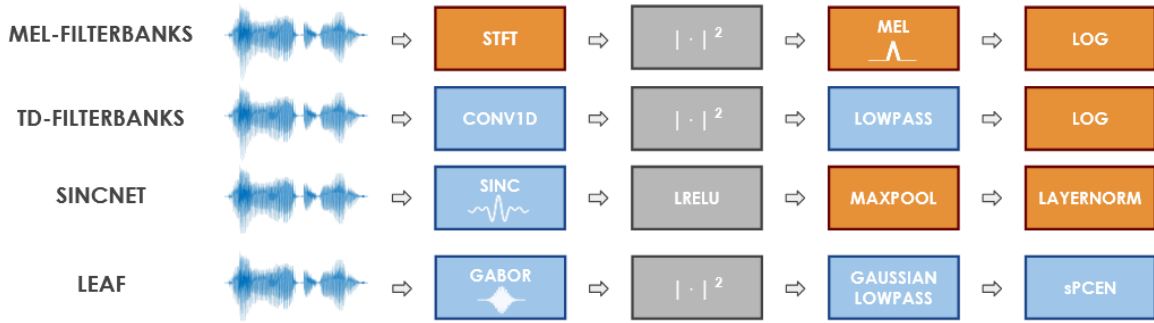


FIGURE 1.9 – Décomposition du calcul par bancs de filtres Mel, par bancs de filtres temporels, de SincNet et de l'interface LEAF. Les cases orange correspondent aux traitements fixes, tandis que les cases bleues représentent ceux apprenables. Les cases grises indiquent les fonctions d'activation. *Source : [25]*

Cette fois-ci, les filtres de la première couche ne sont pas définis par la fonction sinc mais par une fonction de Gabor 1D. Ce type de filtre peut être écrit comme une gaussienne modulée par une sinusoïde :

$$g(t; f, \sigma) = \exp\left(-\frac{t^2}{2\sigma^2}\right) \cdot \cos(2\pi ft), \quad (1.8)$$

où f est la fréquence centrale du filtre et σ la largeur de la gaussienne (contrôle la bande passante). Tout comme dans SincNet, ces filtres sont caractérisés par deux paramètres pouvant être appris par descente de gradient et définissent une forme de filtrage passe-bande. *On peut noter que ce type de convolution est considéré, par une partie importante de la communauté scientifique, comme biologiquement plus réaliste que le concept de convolutions classiques appliquées à la reconnaissance visuelle. Les raisons de cette affirmation étant complexes et en dehors du cadre de ce manuscrit, nous ne nous y attarderons pas davantage [26].*

De plus, au lieu d'utiliser une moyenne glissante sur une fenêtre fixe (comme dans un spectrogramme), LEAF introduit un filtrage passe-bas apprenable basé sur une fenêtre gaussienne. Concrètement, pour chaque canal i , la sortie est donnée par :

$$y_i(t) = (x_i^2 * \phi_{\sigma_i})(t), \quad \text{avec} \quad \phi_{\sigma}(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{t^2}{2\sigma^2}\right), \quad (1.9)$$

où $x_i(t)$ est la sortie du i -ème filtre de Gabor et σ_i (un paramètre appris) contrôle la largeur temporelle du lissage. Ce mécanisme permet d’agréger l’énergie localement dans le temps, tout en laissant le réseau apprendre la durée optimale de lissage, qui peut varier d’un filtre à l’autre.

Enfin, une compression logarithmique est appliquée en sortie, mais de manière inégale entre les filtres. Cette étape, très similaire au PCEN décrit en début de section, constitue une manière plus flexible de paramétrer la compression, tout en restant entièrement différentiable. À l’issue du pipeline, on obtient une représentation bidimensionnelle du signal, définie sur une échelle apprise automatiquement à partir des données, mais dont la forme reste partiellement tributaire de l’initialisation [27].

D’un point de vue computationnel, cette approche est plus coûteuse qu’un spectrogramme classique, sans pour autant garantir une amélioration significative des performances (bien qu’elle constitue, à ce jour, l’une des solutions de *frontend* apprenables les plus performantes). Dans le cadre de cette thèse, nous avons donc choisi d’explorer une autre stratégie, animés du même objectif, mais mieux adaptée à nos contraintes initiales.

1.2.4 Ce qu’il faut en retenir

On retrouve, dans la grande majorité des approches de l’état de l’art, une chaîne de traitement relativement standard : (i) une transformation numérique du signal pour obtenir un spectrogramme 2D ; (ii) une projection sur une échelle de fréquences (souvent MEL) ; (iii) un traitement par un CNN de grande taille pour effectuer la classification.

Quelle que soit la variante considérée dans ce chapitre, toutes ces méthodes restent globalement équivalentes et présentent deux limitations majeures : (i) les réseaux employés exigent d’importantes ressources calculatoires, ce qui empêche leur déploiement en environnement embarqué dans la plupart des cas ; (ii) indépendamment du coût énergétique lié à l’inférence du réseau, les approches requièrent systématiquement un enregistrement numérique et continu du signal audio, ainsi qu’un nombre élevé d’opérations pour chaque transformation. Ce processus impose à l’IA un plancher de consommation énergétique fixe qui est structurellement incompatible avec un fonctionnement globale en ultra-basse consommation.

La reconnaissance acoustique biologique ne fonctionne pas ainsi.

Ce constat motive d’ailleurs de nombreux chercheurs à travers le monde à proposer, tout comme nous, des approches s’inspirant (de manière plus ou moins directe) de ce principe, dans le but de réduire la consommation énergétique. Le chapitre suivant abordera les travaux réalisés autour de ces deux problématiques, qui constitueront les premières pistes aux contributions majeures de cette thèse.

Comment réduire la consommation énergétique de l’IA ?

2.1 Contexte historique

La question de l’efficacité énergétique des réseaux de neurones émerge dès les années 1980–1990, en parallèle des premiers résultats marquants du domaine. Les travaux fondateurs sur les circuits neuromorphiques VLSI analogiques, initiés par Carver Mead et Conway [28], ont montré que des implémentations silicium inspirées de la biologie pouvaient traiter l’information avec une très faible consommation. Les rétines en silicium et capteurs de vision neuromorphiques développés dans cette lignée (notamment avec M. A. Mahowald) [29] sont d’ailleurs les précurseurs des caméras événementielles modernes [30, 31], illustrant le potentiel énergétique des architectures dites *non-von Neumann* [32].

Parallèlement, des techniques d’optimisation des modèles neuronaux numérique apparaissent. Par exemple, l’un des premiers articles sur le sujet, par *LeCun et al.*, introduit une méthode d’élagage supprimant les synapses redondantes dans des réseaux entraînés [33].

Toutefois, entre les années 2000 et début 2010, la majeure partie de la recherche s’oriente plutôt vers des objectifs de performance. Le domaine voit arriver des modèles toujours plus grands, tirant parti de plateformes numériques plus puissantes (GPU/TPU), et reléguant l’efficacité énergétique au second plan.

Il faudra attendre la fin des années 2010 pour que la tendance reparte en sens inverse : l’impossibilité de déployer des réseaux de très grande taille en environnement embarqué, combinée à une demande croissante de dispositif autonome, relance l’intérêt scientifique pour des IAs plus efficaces [34].

2.2 Les méthodes de décimation neuronale

A partir de ce contexte, un large éventail de techniques ont été proposées pour améliorer l'efficacité computationnelle et énergétique des réseaux de neurones numériques. Parmi les plus anciennes et les plus étudiées figurent les techniques d'élagage et de compression de modèles.

2.2.1 L'élagage

L'idée centrale de l'élagage (ou *pruning*) consiste à supprimer les connexions neuronales qui contribuent peu à la sortie du réseau, afin de réduire le nombre d'opérations à l'inférence (et donc la consommation) ainsi que l'empreinte mémoire du modèle.

La principale difficulté (et la raison de la diversité des méthodes) réside dans l'identification des poids pouvant être supprimés sans dégrader significativement les performances. [35, 36] soulignent que l'architecture des réseaux (profondeur, largeur et type des couches) est souvent choisie empiriquement, sans justification théorique autre que le succès de convergence observé à l'entraînement. [37] montre que les réseaux sont fréquemment surparamétrés non pas parce que chaque paramètre est indispensable, mais parce que les modèles plus grands convergent plus aisément vers de bonnes solutions lors de l'apprentissage¹.

Bien que de plus petits modèles puissent, en théorie, atteindre des performances similaires, la très grande dimensionnalité de l'espace d'optimisation rend peu probable statistiquement la découverte de telles solutions avec des méthodes d'entraînement standard. Pour y remédier, de nombreuses techniques d'élagage intègrent une régularisation L1, qui pousse la majorité des poids vers zéro, ou L2, qui limite l'amplitude des poids et réduit le surapprentissage. Des travaux récents ont montré que des cycles itératifs d'élagage et de réentraînement peuvent réduire le nombre de paramètres de près d'un ordre de grandeur sans une perte notable des performances prédictives [39].

2.2.2 La distillation des connaissances

Une approche auxiliaire pour réduire le nombre de paramètres utilisés consiste à compresser le modèle volumineux dans une version plus compacte. Cette stratégie peut être mise en œuvre par la *distillation de connaissances* (ou *knowledge distillation*), introduite par Hinton et al. [40]. Le principe consiste à entraîner un modèle de taille réduite, appelé *élève*, à reproduire la distribution de sorties d'un modèle plus complexe, désigné comme *enseignant*.

Contrairement à un apprentissage direct fondé uniquement sur les données d'entraînement, le modèle élève exploite les représentations internes produites par l'enseignant. Cette approche favorise une meilleure généralisation et oriente la convergence des poids vers une solution connue.

1. Cet effet est à nuancer, car une sur-paramétrisation excessive peut aussi nuire aux performances en augmentant la probabilité de convergence vers de mauvais minima locaux [38].

En outre, plusieurs variantes ont été proposées afin d’enrichir le schéma initial. Parmi elles, l’*alignement de caractéristiques intermédiaires* [41] et le *transfert d’attention* [42] fournissent des signaux d’apprentissage supplémentaires, renforçant la capacité du modèle élève à exploiter efficacement l’information issue de l’enseignant.

2.2.3 Les encodeurs

Une autre méthode efficace pour réduire l’espace neuronal mobilisé consiste à recourir à des architectures de type *encodeur*, capables de transformer des données d’entrée de haute dimension en une *représentation latente* compacte. Les auto-encodeurs constituent une mise en œuvre courante de ce principe : ils reposent sur une paire encodeur-décodeur, où l’encodeur compresse l’entrée tandis que le décodeur tente de la reconstruire. Après l’entraînement, le décodeur peut être écarté, ne conservant que l’encodeur afin de fournir des entrées compressées à un classificateur ou à tout autre modèle en aval [43]. Les auto-encodeurs variationnels (VAEs) [44], de même que ceux fondés sur l’apprentissage contrastif, produisent des espaces latents structurés qui conservent les informations sémantiques essentielles, tout en réduisant de manière significative la dimensionnalité des données.

Au-delà de la réduction du volume de données, ces techniques améliorent généralement la robustesse au bruit et optimisent la pertinence des représentations pour la tâche visée, contribuant à l’efficacité globale du système.

2.2.4 La quantification

Ces méthodes de réduction structurelle peuvent également être combinées avec une quantification des poids en mémoire. La quantification diminue la taille du modèle ainsi que la charge computationnelle en représentant les poids et les activations à l’aide de formats de précision plus faible. Elle exploite le fait que les opérations entières ou en virgule fixe sont nettement moins coûteuses que les opérations en virgule flottante sur le matériel moderne. Concrètement, au lieu d’une représentation classique en virgule flottante 32 bits, les réseaux peuvent ainsi utiliser des formats en 16 bits, 8 bits, voire 4 bits ou même 1 bit [45]. La quantification *post-training* est simple à appliquer, mais la quantification consciente de l’entraînement (QAT) fournit généralement de meilleurs résultats [46].

D’autres techniques du même genre, telles que le partage de poids et la factorisation en basse-rang, sont également répandues : la première regroupe des paramètres similaires afin de réutiliser l’espace de stockage, tandis que la seconde décompose les matrices de poids afin de réduire leur dimensionnalité [47, 48].

2.3 L’optimisation matérielle

Les modèles quantifiés ouvrent la voie à des gains supplémentaires lorsqu’ils sont optimisés conjointement avec le matériel qui les exécute [49]. Un exemple emblématique est le développement des accélérateurs d’intelligence artificielle, tels que le *Tensor Processing Unit* (TPU) de Google, dont l’architecture repose sur les opérations fondamentales des réseaux neuronaux : des produits scalaires massivement parallélisés, implémentés sous forme de

multiplications matricielles et de convolutions. En combinant une arithmétique à précision réduite (par exemple, en entiers 8 bits) avec une architecture en réseau systolique, ces dispositifs atteignent une efficacité énergétique pouvant être 30 à 80 fois supérieures à celles des CPU et GPU conventionnels pour des tâches d’inférence comparables [50].

De plus, cette optimisation matérielle pour un réseau donné ne se limite pas à un travail sur la précision des poids. Les réseaux clairsemés issus de l’élagage, par exemple, peuvent être déployés sur du matériel exploitant des formats de matrices creuses, évitant ainsi les multiplications par zéro et réduisant significativement la consommation énergétique [51].

En somme, ces approches de *co-design* matériel-logiciel considèrent le réseau neuronal et le processeur comme un système unifié, optimisé conjointement. Ce paradigme conduit fréquemment à des gains d’efficacité importants, comme l’illustrent les accélérateurs spécialisés (TPUs, *Neural Compute Stick* d’Intel, *Deep Learning Accelerator* de NVIDIA) ainsi que plusieurs prototypes de recherche, à l’image de la puce Eyeriss du MIT dédiée aux CNN [52]. En adaptant l’algorithme au matériel (et réciproquement), il est possible d’obtenir des gains d’efficacité inaccessibles avec les plateformes de calcul généralistes [53], on parle alors d’implémentation ASIC (*Application-specific integrated circuit*).

Néanmoins, cette approche demeure souvent difficile à déployer en pratique, car elle nécessite un contrôle total sur la conception algorithmique et matérielle, ce qui restreint son implémentation à des scénarios standards proposés par les entreprises privées détenant les puces accélératrices et les outils de conversion associés. Dans la suite du manuscrit, nous reviendrons sur ce levier d’optimisation afin de viser une efficacité supérieure à celle établie ici ; toutefois, par défaut, il ne sera pas pris en compte dans l’implémentation des PrNNs, que nous voulons volontairement généraliste.

2.4 Les nouvelles couches neuronales

Malgré l’utilisation de méthodes d’élagage et de compression, associées à la quantification et à l’emploi d’accélérateurs matériels dédiés, les performances restent limitées par les mécanismes opérationnels fondamentaux des réseaux de neurones.

Historiquement, les rares ruptures algorithmiques ayant réellement permis de dépasser ce plafond de verre reposent sur l’introduction de nouveaux modes de calcul bio-inspirés. Ce fut le cas lors de la création des couches neuronales majeures, telles que les convolutions, à l’origine des CNN [54], et les réseaux de neurones récurrents (RNN) [55], ensuite stabilisés pour l’apprentissage par les LSTM. Ces progrès computationnels sont cependant à relativiser lorsque l’analogie biologique est peu marquée. Concrètement, l’innovation des Transformers [56] relève de ce cas : malgré des gains de performance indiscutables, ceux-ci s’accompagnent d’une forte inflation du nombre de paramètres et des ressources computationnelles, tout en s’éloignant des mécanismes d’attention observés en biologie.

L’introduction de ces nouvelles couches a suscité un sous-domaine entier de recherche en architectures neuronales, visant à optimiser l’agencement des couches et à en développer de nouvelles, moins coûteuses en calcul, mais aux performances comparables (par exemple,

les GRU en alternative aux LSTM [57, 58]).

La recherche automatisée d’architectures (NAS) pousse plus loin cette optimisation en automatisant l’exploration de l’espace de conception sous des contraintes explicites de latence ou d’énergie. Des techniques fondées sur l’apprentissage par renforcement [59] et sur les algorithmes évolutionnaires [60] permettent d’ajuster précisément les architectures aux limitations matérielles. Plus récemment, des cadres multi-agents adossés à des LLM automatisent la génération, l’orchestration et l’évaluation d’architectures candidates, élargissant l’espace de recherche au-delà des catalogues existants et révélant parfois des configurations inédites [61].

2.5 Les architectures neuromorphiques

[62] montre que la puissance cumulée nécessaire au cerveau et aux organes sensoriels est inférieure de quatre ordres de grandeur ($\div 10\,000$) à celle de systèmes électroniques CMOS classiques à puissance de calcul équivalente.

Compte tenu de cet écart massif, il paraît évident qu’un véritable impact sur la consommation énergétique ne pourra venir que d’un basculement fondamental des méthodes de calcul vers des méthodologies bio-inspirées [63]. Dans cette perspective, de nombreux chercheurs et industriels s’orientent vers des architectures neuromorphiques telles que des systèmes sur puce neuromorphiques (NeuSoC) et des réseaux de neurones impulsionnels (SNN). Plusieurs synthèses récentes anticipent des gains d’efficacité de plusieurs ordres de grandeur [64, 65].

Des systèmes tels qu’*IBM TrueNorth* et *Intel Loihi* illustrent une co-conception algorithmes–matériel en mettant en œuvre des neurones impulsionnels et des architectures silicium asynchrones, massivement parallèles [66, 67]. Ces puces exploitent un calcul parcimonieux, déclenché par des impulsions, ce qui se traduit par des économies d’énergie substantielles sur des tâches pilotées par les événements [68]. Couplées à des capteurs événementiels, elles permettent des chaînes de vision et de traitement entièrement neuromorphiques, rapprochant le flux d’information de celui observé en biologie [69].

Bien que prometteuses, ces architectures mettent le plus souvent en œuvre l’asynchronisme impulsif via des mécanismes de routage numérique, ce qui s’apparente plus à une émulation « logicielle » (et non matérielle) du comportement asynchrone. Cette approche est pertinente pour des puces comportant un grand nombre de neurones et de cœurs, car le surcoût d’une asynchronie gérée numériquement s’amortit à l’échelle du système. En revanche, pour des conceptions de petite taille avec un nombre limité de neurones, le bilan énergétique peut devenir moins favorable. Dans ces cas, des conceptions entièrement analogiques ou des architectures impulsives asynchrones, souvent à base de *mémristors* ou de *circuits asynchrones spécialisés*, sont activement explorées ; elles restent toutefois peu accessibles, et les résultats demeurent insuffisamment mûrs pour un déploiement à grande échelle à ce jour.

2.6 Résultats concrets

A ce stade, l'ensemble des méthodes discutées a déjà été mis en oeuvre dans la littérature à travers des exemples concrets de classification sonore. Le plus souvent, ces travaux se déroulent dans des environnements contrôlés (en intérieur) et pour des tâches de complexité modérée, en particulier le *keyword spotting* (KWS).

Le tableau 2.1 présente une synthèse de l'état de l'art sur le corpus *Google Speech Commands* [70], composé d'énoncés de mots-clés d'environ 1 s. *Les valeurs rapportées dépendent fortement des conditions d'implémentation (matériel, fréquence, tension, bibliothèques, protocole de test) et des valeurs reportées par les auteurs ; elles doivent être interprétées avec prudence.*

En particulier, les plateformes Raspberry Pi intègrent par défaut de nombreux périphériques à la consommation de repos élevée et ne visent pas l'ultra-basse consommation..

TABLE 2.1 – État de l'art des approches de reconnaissance sonore sur le corpus *Google Speech Commands* (hors ASIC). Consommation hors microphone. 1 inf./s considérée.

Article	Méthode d'acquisition et réseau	Caractéristiques Matériel	Performance (KWS) ²			
			Latence (ms)	Puissance moyenne (mW)	Nb. de Classes	Accuracy (%)
2018 Hello Edge [71]	MFCC DS-CNN 8-bit (S ver.) 39 KB 5.4 MMACs	STM32F746 ARM Cortex-M7 216 MHz, 3.3V	12	14.90 ($\approx 2.9 + 12.0$)	10	94.4
2021 [72] Ulkar-Okman	1D Conv CNN 8-bit QAT 420 KB	MAX78000 CNN Accelerator	3.5	1.50 ($\approx 1.25 + 0.25$)	10	96.3
2021 TsetlinM [73]	Binarised MFCC Tsetlin Machine 240 Clauses/class	STM32F746 Arm Cortex-M7 216 MHz, 3.3V	730 (160+570)	14.29 (2.14 + 12.15)	8	88.0
2021 [74] Cerutti et al.	Mel-Spec (3.2 s) BNN 884 MMACs	RISC-V GAP8 XNOR-Accel 150 MHz, 1.0V	511	28.18 (2.64 + 25.54)	28 ³	77.9 ³
2022 Edge-CRNN [75]	LFBE-Delta CRNN	RaspberryPi 3B+ ARM Cortex-A53 1.2 GHz	200	≈ 4000	12	97.9
2022 BiFSMN [76]	Mel-Spec Wavelett Trans. Binarised NN	RaspberryPi 3B+ 1.2 GHz	80	≈ 3500	12	95.0
2022 Wang-Li [77]	Mel-Spec CNN-one-fstride4 int16 + SIMD 468 KB 476 kMACs	STM32F746 216 MHz WM8978 chip OS :FreeRTOS	15.6	20.50 ($\approx 3.00+17.50$)	6	90.0
2023 TinyML [78]	MFCC ResNet15 10-shot FAR=5% 235 MMACs	GAP9 240 MHz, 0.65V	9.21	25	12	80.0

²À titre de comparaison, les objectifs de ce manuscrit fixent pour ce type de tâche une puissance moyenne cible de **0.25 mW** pour une précision **supérieure à 90 %**.

³Réalisé sur le dataset https://data.vision.ee.ethz.ch/cv1/ae_dataset/

2.7 Vers la création des PrNNs

Les contributions discutées dans les sections précédentes partagent une limitation commune : elles se concentrent uniquement sur l’optimisation de la consommation énergétique de l’unité de traitement neuronal, tandis que la numérisation, la mise en mémoire tampon, et les prétraitements discutés (transformée de Fourier) introduisent des surcoûts énergétiques non négligeables dans des systèmes en régime continu (dit *always-on*). Ce phénomène éclaire pourquoi, malgré l’intégration de ces techniques, les meilleurs modèles de l’état de l’art ne parviennent pas à réduire leur consommation en dessous du milliwatt.

À l’inverse, les systèmes biologiques ont optimisé non seulement leur traitement cognitif (le cerveau), mais aussi leurs organes sensoriels et les voies neuronales servant à leurs liaisons. Cette évolution conjointe, spécifique à un ensemble de tâches et pour un environnement donné, est ce qui rend les systèmes biologiques plus efficaces que nos IAs traditionnelles.

Il existe un certain nombre de travaux récents qui ont pris acte de cette réalité et qui mettent en oeuvre une captation sensorielle analogique dédiée à alimenter à posteriori une utilisation cognitive numérique [62, 79–81, 81–83] (ou impulsionnelle dans de plus rares cas [84, 85]). Dans notre contexte, les méthodes proposées reposent sur une extraction analogique des fréquences d’intérêt au moyen de filtres matériels, ainsi que (parfois) des traitements neuronaux analogiques, ce qui, nous le démontrerons, se révèle plus économe en énergie. En un sens, cette captation est une analogie directe au fonctionnement des organes biologiques et nécessite des puissances énergétiques bien en dessous du reste de l’état de l’art.

Néanmoins, les structures proposées sont souvent figées par leurs concepteurs sur la base d’une connaissance très précise du domaine d’application (par exemple la sélection de bandes fréquentielles critiques), relevant davantage de l’application de solutions connues que d’un véritable mécanisme d’adaptation ou d’évolution. Notre idéal n’est pas dans ce genre d’architecture mais se rapproche plutôt de l’envie originelle des travaux de M. Ravanelli & Y. Bengio, qui était d’apprendre de manière automatique la meilleure représentation sonore possible pour la tâche donnée.

C’est à partir de cette vision holistique que nous est apparu le concept de *Réseaux neuronaux primitifs* (PrNN) [2, 3] : des architectures biomimétiques conçues pour des tâches de reconnaissance *always-on*, telles que la classification acoustique, sous des contraintes de puissance ultra-faible. Pour parvenir au paradigme associé, décrit en Partie II, nous nous inspirons de la co-évolution biologique du cerveau et des organes sensoriels afin de proposer une approche de co-conception qui adapte conjointement l’architecture neuronale et l’interface sensorielle.

Le fonctionnement biologique en acoustique

Tous les mammifères possèdent un organe dédié à l'audition : l'oreille. Dans ce chapitre, nous décrivons le fonctionnement de l'oreille humaine et son lien avec le cerveau qui, à de rares exceptions près, est très similaire chez les autres espèces.

L'oreille humaine est communément divisée en trois sections : externe, moyenne et interne ; qui coopèrent pour capter, transmettre et analyser les sons. Chacune contribue différemment au traitement du signal.

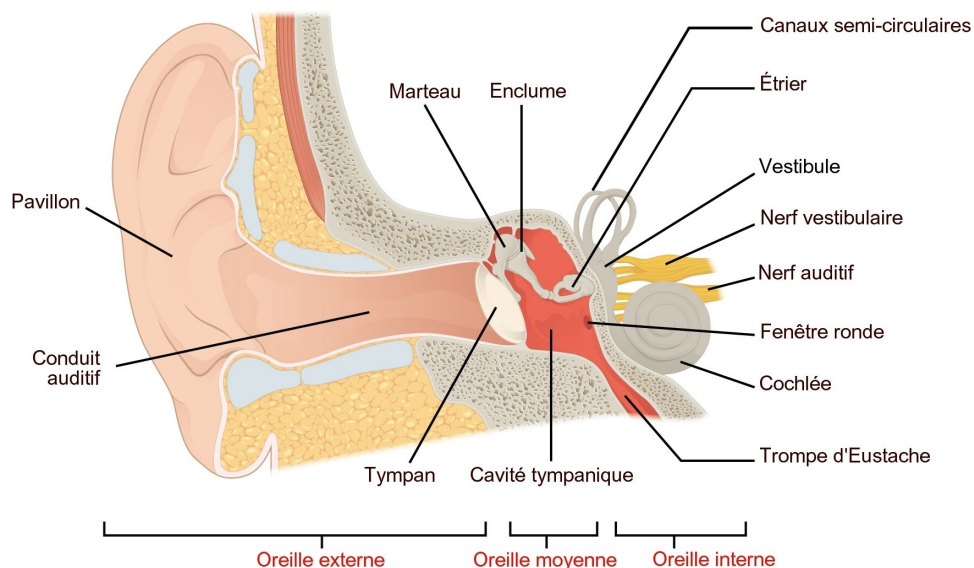


FIGURE 3.1 – Structure de l'oreille (coupe frontale à travers le côté droit du crâne)
D'après l'illustration CC By OpenStax via Wikimedia Commons

3.1 L'oreille externe

L'oreille externe se compose de deux parties distinctes : le pavillon auriculaire (*pinna*), visible à l'extérieur, et le conduit auditif reliant ce pavillon au tympan.

La forme du pavillon auriculaire est spécialement conçue pour capturer et orienter les ondes sonores vers le conduit auditif. Sa structure en coquille contribue à concentrer et à amplifier certains sons dans des gammes de fréquences précises, améliorant ainsi la sensibilité de l'oreille à ces fréquences particulières et éliminant celles de bruits parasites.

Étudier cette sélection est complexe, car la forme et la position de chaque oreille varient d'une personne à l'autre. La relation entre la source sonore et les ondes binaurales perçues, connue sous le nom d'Head Related Transform Function (HRTF) [86–88], démontre l'importance de la forme et de la position de l'oreille dans la localisation des sources sonores. La non-symétrie des pavillons induit une collecte sonore inégale et direction-dépendante [89]. Elle rompt l'éventuelle symétrie spectrale entre les deux oreilles (voir Fig. 2), fournissant au système auditif des indices suffisants pour estimer, dans le plan sagittal, la position 2D d'une source. En combinant ces indices spectraux monauraux avec les indices interauraux (ITD/ILD), le cerveau parvient à une localisation pleinement tridimensionnelle malgré la présence de seulement deux oreilles.

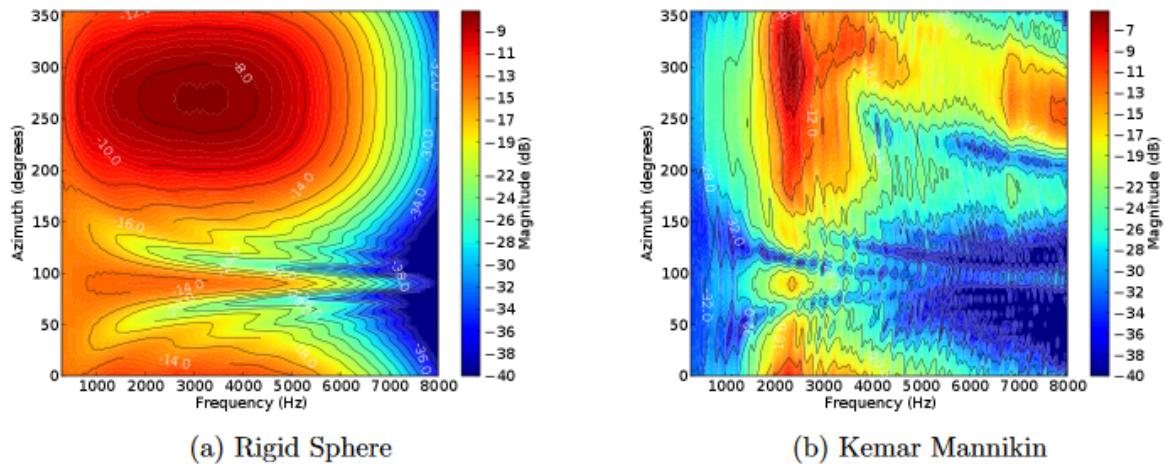


FIGURE 3.2 – HRTF de l'oreille gauche pour (a) un modèle de sphère rigide et (b) des mesures sur mannequin *KEMAR* ($\approx$ une oreille humaine moyenne).

Données expérimentales d'après Gardner et al. (1994), reproduites depuis [89].

Les ondes sonores captées par le pavillon sont guidées à travers le conduit auditif jusqu'au tympan, une membrane marquant la frontière entre l'oreille externe et l'oreille moyenne. Sa principale fonction est de vibrer pour transmettre les ondes sonores provenant de l'extérieur.

D'un point de vue acoustique, le conduit auditif agit comme un tube de résonance. La première fréquence de résonance se situe autour de 3 kHz, fréquence pour laquelle la longueur d'onde sonore dans l'air (10 cm) est environ quatre fois supérieure à celle du

conduit (2,5 cm) [90]. Grâce à cette structure, la pression des oscillations est augmentée d'environ 10 dB entre 2 et 5 kHz, ce qui concorde avec une meilleure sensibilité humaine dans cette bande de fréquence.

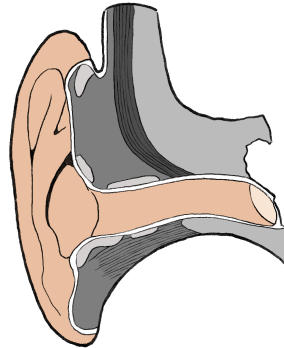


FIGURE 3.3 – Schéma de l'oreille externe.

En revanche, la sensibilité diminue de manière significative dans les fréquences plus basses, car la longueur des ondes devient importante comparativement à la taille du conduit auditif.

3.2 L'oreille moyenne

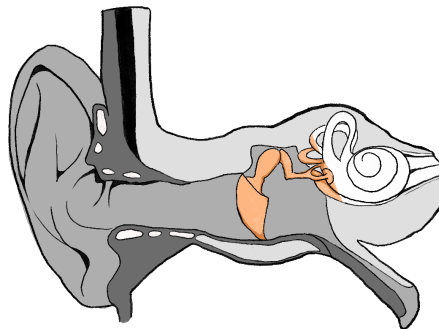


FIGURE 3.4 – Schéma de l'oreille moyenne.

Transmission sonore

L'oreille moyenne commence à partir du tympan et est composée de trois petits os, le marteau (*malleus*), l'enclume (*incus*) et de l'étrier (*stapes*) qui est relié à la fenêtre ovale ; marquant le début de l'oreille interne. Ceux-ci sont les trois plus petits des 206 os du corps humain.

Lorsque les ondes sonores atteignent le tympan, les variations de pression de l'air font bouger le tympan en va-et-vient, et les osselets transmettent cette vibration à la fenêtre ovale de l'oreille interne.

L'oreille interne étant remplie d'un liquide aqueux, les sons qui sont transmis dans ce milieu sont naturellement atténués. Cela est dû au principe de réfraction, qui revoie une partie des ondes à la frontière entre deux milieux distinct. L'impédance sonore de l'eau étant bien plus importante que celle de l'air ($Z_{air} = 430 \text{ Pa.s.m}^{-1}$; $Z_{eau} = 1.5 * 10^6 \text{ Pa.s.m}^{-1}$), les oscillations de pression sont atténuées d'environ 30 dB lors de leur transmission à l'oreille interne [91].

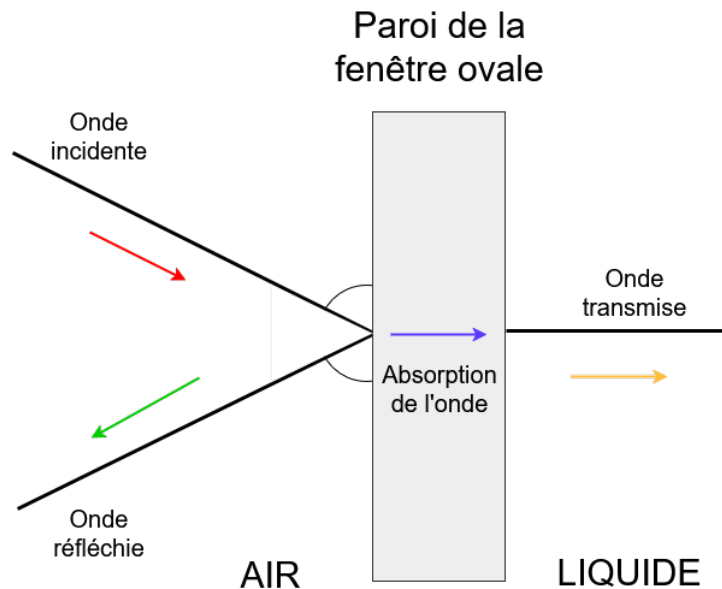


FIGURE 3.5 – Schéma de la réflexion acoustique entre différents milieux

Cette atténuation est compensée par la chaîne ossiculaire, qui assure une adaptation d'impédance et augmente la pression transmise. À l'interface tympan-marteau, l'amplitude de l'oscillation est plus grande pour une force plus faible ; à l'extrémité opposée (étrier-fenêtre ovale), l'amplitude diminue tandis que la pression augmente [92].

L'amplification, de l'ordre de 26 dB, résulte de deux mécanismes complémentaires : (i) l'effet de levier du couple marteau-enclume et (ii) le rapport de surfaces entre le tympan et la fenêtre ovale. Comme la pression p est donnée par $p = F/S$, l'augmentation de la force et la diminution de la surface se conjuguent pour accroître la pression à l'entrée de la cochlée.

Un réflexe de protection intervient à forts niveaux d'intensité sonore : chez l'humain, c'est principalement la contraction du muscle stapédien qui rigidifie la chaîne ossiculaire et limite la transmission vers l'oreille interne.

L'anatomie de l'oreille moyenne varie selon les espèces. Chez les reptiles, amphibiens et oiseaux, un seul osselet (la *columelle*), relie le tympan à la fenêtre ovale [93]. Dans ce cas, l'amplification repose surtout sur le rapport de surfaces des membranes. Chez les reptiles et les amphibiens, la bande de sensibilité maximale se situe autour de quelques kilohertz, tandis qu'elle atteint généralement 8–13 kHz pour les oiseaux [94]. À l'inverse, certaines espèces de chiroptères présentent une sensibilité auditive allant de 20 à 150 kHz.

Équilibrage de la pression

Lorsque la pression extérieure varie (altitude, plongée, etc.), un gradient peut apparaître de part et d'autre du tympan. Équilibrer ces pressions est essentiel pour préserver la mobilité de la membrane et éviter toute lésion. La trompe d'Eustache (ou tube auditif), conduit reliant la caisse du tympan au nasopharynx, assure cette égalisation : normalement fermée, elle s'ouvre transitoirement¹ pour ventiler l'oreille moyenne et égaliser les pressions, puis se referme, protégeant l'oreille des variations rapides et des reflux.

À l'interface avec l'oreille interne, l'équilibrage est assuré par la compliance de la fenêtre ronde : les déplacements du pied de l'étrier à la fenêtre ovale compriment la périlymphe, et la fenêtre ronde, située en dessous, se déplace en sens inverse, jouant un rôle de soupape de décharge. Ce couplage évite toute surpression statique dans l'oreille interne (voir Fig. 3.1).

3.3 L'oreille interne : le système vestibulaire

Cette section constitue une brève digression : le système vestibulaire ne contribue pas directement à l'audition, mais il est co-localisé avec la cochlée au sein de l'oreille interne. Il est donc pertinent d'en rappeler brièvement l'organisation et sa fonction dans un chapitre d'état de l'art consacré à l'oreille.

Le système vestibulaire renseigne le cerveau sur les accélérations angulaires et linéaires de la tête ainsi que sur son orientation par rapport à la gravité.

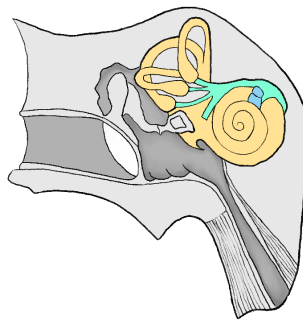


FIGURE 3.6 – Schéma de l'oreille interne. On y voit le système vestibulaire ainsi que la cochlée.

Les canaux semi-circulaires

Visuellement, ce système est composé de trois canaux semi-circulaires : antérieur, postérieur et latéral, disposés à peu près orthogonalement et servant à la détection des accélérations angulaires de la tête. Ces canaux sont remplis d'un liquide, nommé l'endolymphe (ou *liquide de Scarpa*). Aux extrémités de chaque canal se trouve une *ampoule* osseuse

1. Le plus souvent lors de la déglutition, du bâillement ou de la mastication.

(il en existe quatre au total) renfermant une *crête ampullaire* (*crista ampullaris*) coiffée d'une *cupule* gélatineuse, dans laquelle baigne l'extrémité de cellules stéréo-cillées.

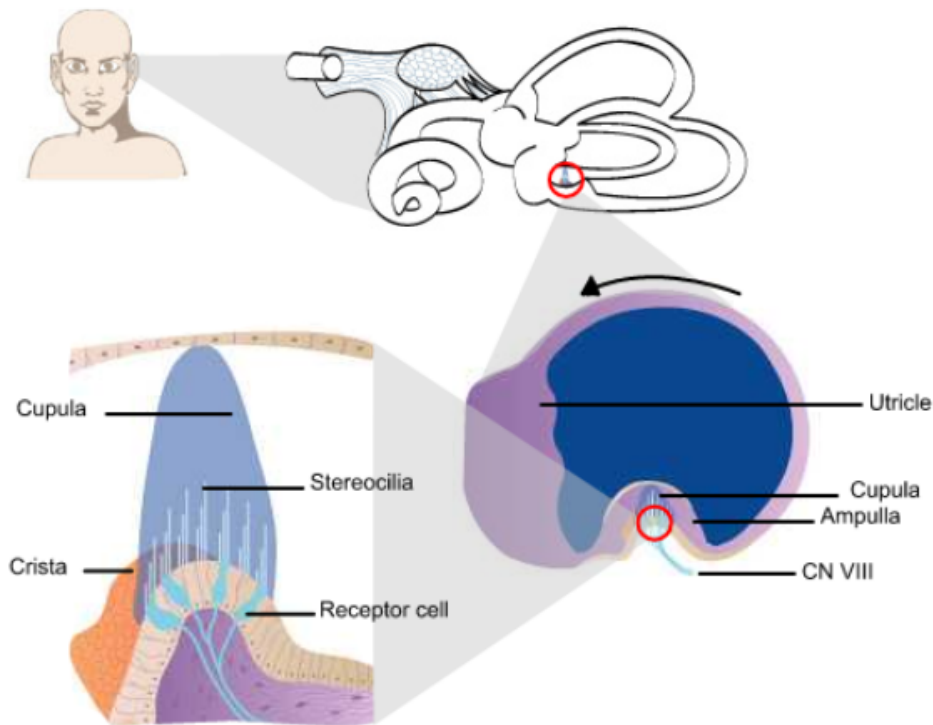


FIGURE 3.7 – Schéma structurel des canaux semi-circulaires de l'oreille interne. Un focus est apporté sur la composition des crêtes ampulaires.

Source : <https://nba.uth.tmc.edu/neuroscience/m/s2/chapter10.html>

Lors d'une rotation, l'endolymphe se retrouve désynchronisée au sein des canaux. L'inertie induit une déflexion de la cupule qui plie les stéréocils, ce qui entraîne une stimulation ou une inhibition nerveuse en ouvrant/fermant des canaux potassiques.

Le fait d'avoir deux oreilles réparties symétriquement par rapport au plan sagittal implique que le mouvement de l'endolymphe est opposé en direction pour chaque canal opposé. Les canaux fonctionnent par paires pour permettre de détecter la vitesse angulaire. Un tour de tête excite les récepteurs d'une ampoule et inhibe les récepteurs de l'ampoule opposée, permettant au cerveau de déterminer le sens de rotation.

Dans la nature, on observe une corrélation entre la taille de ces canaux et l'agilité de l'animal. Plus l'espèce est rapide et agile, plus ces canaux sont grands et vis-versa.

La saccule et l'utricule

Le système vestibulaire est aussi pourvu de deux organes otolithiques : la saccule et l'utricule (Fig.3.8) qui permettent de déterminer la position de la tête par rapport à la gravité terrestre et les accélérations linéaires. Leurs récepteurs, appelés macules (*maculae*), sont des regroupements de cellules cillées accrochées à de petits cristaux (principalement composés de calcium et de carbone), nommés otoconies (*otoconia*).

Ces cristaux plient les cils dans le sens de la gravité ou de l'accélération linéaire, entraînant une stimulation [95]. Ces deux organes sont disposés avec un angle de 90° sur la paroi de l'oreille interne (utricle plutôt horizontal, saccule plutôt vertical), ce qui assure une sensibilité directionnelle complémentaire.

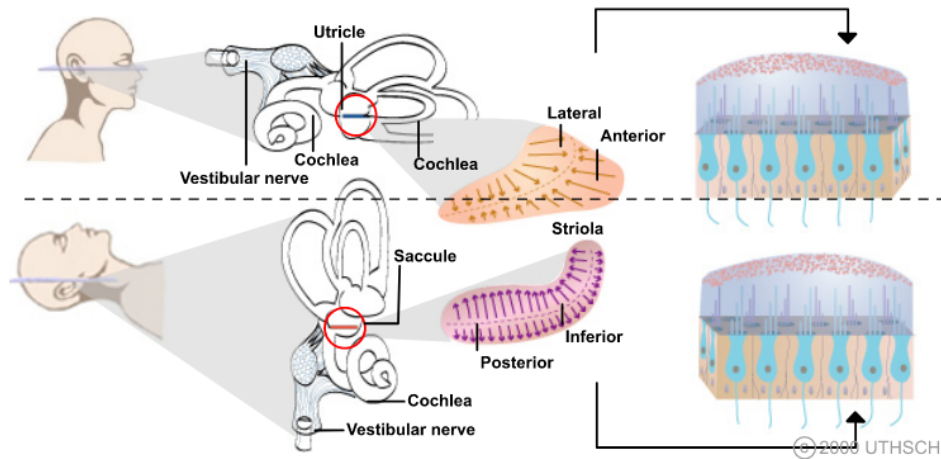


FIGURE 3.8 – Représentation de la saccule et de l'utricle.

Source : <https://nba.uth.tmc.edu/neuroscience/m/s2/chapter10.html>

Terminaisons nerveuses

Toutes les terminaisons nerveuses du système vestibulaire sont regroupées avec celles de la cochlée pour former le nerf vestibulocochléaire (VIII). La majorité de ces connexions nerveuses se projettent ensuite vers l'un des quatre noyaux vestibulaires voisins (supérieur, médial, latéral, inférieur) situés entre le *bulbe* (moelle allongée) et le *pont de Varole*. Les quelques connexions restantes passent directement par le pédoncule cérébelleux inférieur pour atteindre le cervelet (notamment le lobe flocculo-nodulaire) [96].

Certains problèmes peuvent apparaître lorsqu'un conflit survient entre les informations vestibulaires et visuelles transmises au cerveau. Des études ont mis en évidence ces relations et permis d'identifier les conditions responsables du mal des transports [97, 98].

Bien que le vestibule ne traite pas les sons, des sons très intenses peuvent activer les organes otolithiques (phénomène de Tullio), entraînant vertiges et instabilité chez la personne. D'une manière générale, les cellules ciliées vestibulaires se régénèrent très peu et les otoconies peuvent se détacher (p. ex. VPPB) ; ces altérations sont souvent responsables de troubles persistants de l'équilibre.

3.4 L'oreille interne : La cochlée

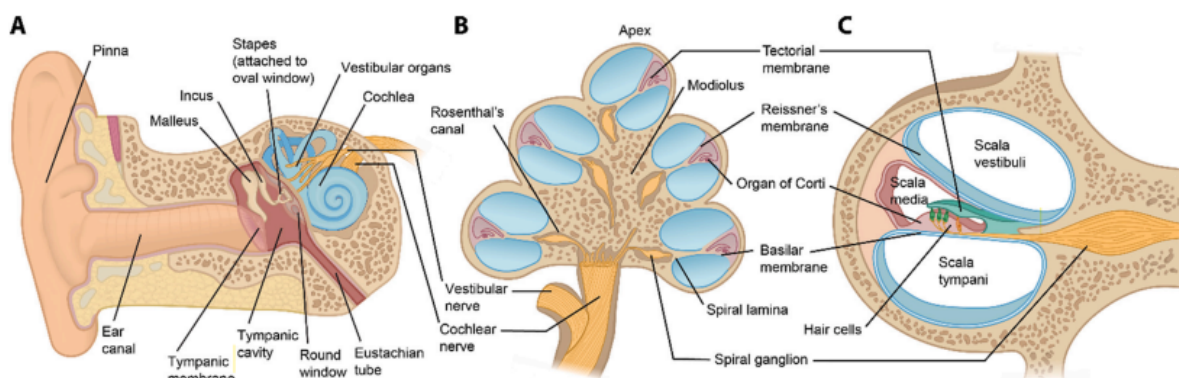


FIGURE 3.9 – Anatomie de l'oreille humaine. **A.** Anatomie de l'oreille humaine, y compris les organes auditifs et vestibulaires. **B.** Illustration schématique d'une coupe cochléaire montrant comment le canal cochléaire s'enroule autour du modiolus contenant les neurones auditifs. **C.** Vue agrandie d'une coupe transversale d'une spire cochléaire. Notons que les rampes vestibulaire et tympanique connectent au sommet de la cochlée.

Figure créée avec BioRender.com et traitée ultérieurement par les auteurs de [91].

Structure anatomique

La cochlée est une structure en forme de coquille enroulée, située dans l'oreille interne, dont le rôle principal est de convertir les vibrations sonores en signaux électriques interprétables par le cerveau. Déroulée, elle mesure 35 mm de long pour une section transversale de 2 mm^2 [91]. La cochlée est subdivisée en trois chambres : le canal vestibulaire (*Scala vestibuli*), le canal tympanique (*Scala tympani*) et le canal cochléaire (*Scala media*). Le canal cochléaire, est délimité par la *membrane Tectoria* qui recouvre l'*organe de Corti*, par la membrane vestibulaire (aussi appelée membrane de Reissner) et la strie vasculaire ; une boucle capillaire située sur la paroi externe du canal. L'organe de Corti repose sur la membrane basilaire, située du côté du canal tympanique. Le ganglion spiral et les fibres nerveuses sont reliés aux cellules ciliées de cet organe et représentés en jaune sur la figure 3.9.

Les canaux vestibulaire et tympanique sont remplis d'un liquide nommé la *périmylphe*. Ce liquide, très proche en consistance du liquide cébrospinal, est une substance visqueuse, riche en ion sodium (Na^+) et faible en potassium (K^+).

A l'inverse, le canal cochléaire est rempli d'*endolymphe*, un fluide riche en potassium (K^+), faible en sodium (Na^+) et presque dénué de calcium (Ca^{2+}). Cette différence du niveau de potassium entre les deux fluides est régulée par la strie vasculaire et permet le bon fonctionnement des cellules ciliées de l'organe de Corti.

Le mécanisme de propagation des ondes sonore est le suivant : lorsque l'étrier fait vibrer la fenêtre ovale, l'onde se propage dans le canal vestibulaire puis passe par le canal cochléaire jusqu'à faire vibrer la membrane basilaire du canal tympanique. La surpression générée dans le canal tympanique est ensuite évacuée par la fenêtre ronde située entre la cochlée

et l'oreille moyenne. Plus un son est fort, plus l'amplitude de la vibration de la membrane basilaire sera élevée. Cette vibration est inégale le long de la membrane, chaque zone réagit à des fréquences de résonances différentes et disposées selon une échelle logarithmique le long de la cochlée.

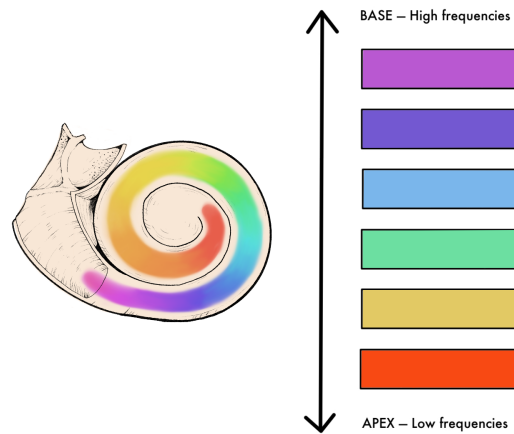


FIGURE 3.10 – Sensibilité fréquentielle le long de la cochlée.

Le canal cochléaire est l'élément central de la détection acoustique, il est composé de nombreuses cellules ciliées qui reposent sur la membrane basilaire (voir Figure 3.11). Ces cellules sont responsables des stimulations nerveuses en réponse à une vibration sonore. Il en existe deux types : les cellules ciliées **internes** (CCI) et les cellules ciliées **externes** (CCE) ; ces cellules sont placées en ligne le long de la cochlée mais sont de structure inégales. A la base, les cils sont rigides et court tandis qu'à l'extrémité (l'apex), ils sont souple et long.

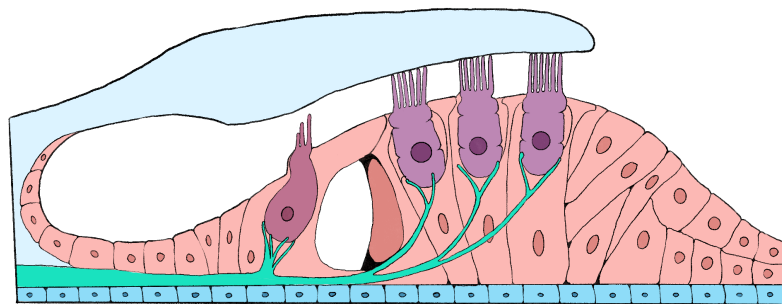


FIGURE 3.11 – Coupe transversale du canal cochléaire. A gauche, une cellule ciliée interne. A droite, trois cellules ciliées externes (correspondant aux trois rangées physique le long de l'organe).

Les CCIs sont principalement responsables de la stimulation nerveuse afférente, (autrement dit, ils transmettent l'information fréquentielle au cerveau) tandis que les CCEs

servent en réalité à modifier le son et permettent d'affiner la sélection fréquentielle (autrement dit, ils modulent le mouvement naturel de la membrane basilaire).

Nous développerons précisément le fonctionnement de ces deux cellules dans leurs sections dédiées, plus tard dans ce chapitre.

Psychoacoustique

L'oreille humaine réagit de manière logarithmique à la pression sonore, généralement exprimée en décibels (dB SPL). La référence utilisée est de $20 \mu\text{Pa}$, correspondant au seuil minimal de perception auditive (0 dB SPL), soit une vibration du tympan de l'ordre du picomètre. À l'autre extrémité de l'échelle, l'oreille peut supporter jusqu'à environ 20 Pa (soit 120 dB SPL).

La sensibilité auditive s'étend sur une très large plage fréquentielle (de 20 à 20 000 Hz) et permet de distinguer des variations de fréquence de l'ordre de 0.2% [99]. Cette capacité n'est cependant pas uniforme : elle est maximale dans la zone de la parole (entre 1500 et 4000 Hz), puis se dégrade progressivement de manière exponentielle aux fréquences plus élevées. Pour mieux rendre compte de cette inhomogénéité perceptive, une nouvelle échelle psychoacoustique fut introduite en 1937 : l'échelle *Mel*. La figure 3.12 illustre la relation entre les mels et les hertz. On y observe une concentration plus importante de points dans les basses fréquences (jusqu'à environ 5 kHz), reflétant la plus grande précision de discrimination auditive dans cette zone.

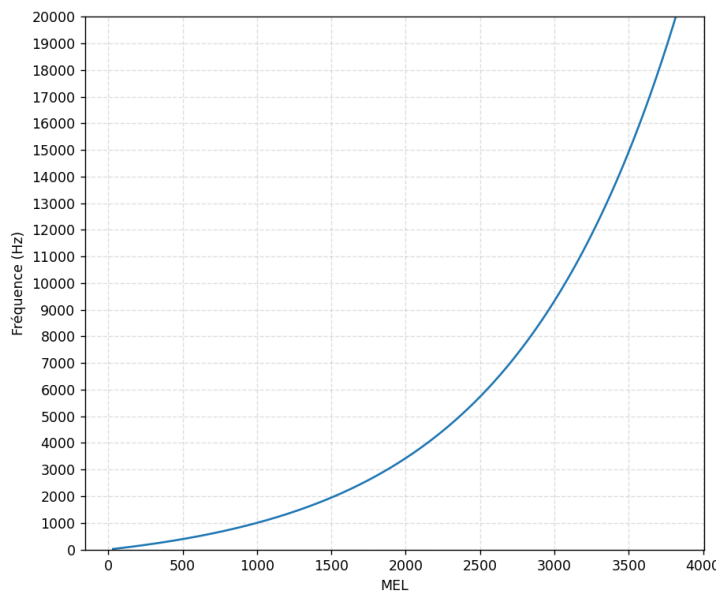


FIGURE 3.12 – Répartition fréquentielle le long de l'échelle MEL. La fonction associée est $f = 700 \times (e^{\frac{m}{1127}} - 1)$.

Cette sélectivité fréquentielle résulte de l'action conjointe de plusieurs mécanismes cochléaires (détaillés dans la prochaine sections), parmi lesquels : les réglages électrochimiques des cellules réceptrices, les variations de résonance mécanique des cellules ciliées, ainsi que les rétroactions actives liées à leur dépolarisation.

Par ailleurs, la perception de l'intensité d'un son pur peut être altérée par la présence d'un autre son [100]. Bien qu'il puisse sembler défavorable à la détection auditive, ce phénomène, connu sous le nom de *masquage*, joue un rôle fondamental dans le traitement des signaux complexes. En effet, le masquage, de nature non linéaire, affecte surtout les composantes faibles en fréquence et en amplitude. Ce mécanisme contribue ainsi à améliorer le rapport signal/bruit (SNR) en supprimant des éléments parasites, et pourrait renforcer l'intelligibilité de la parole en augmentant le contraste entre signaux pertinents et bruit ambiant [101]. Toutefois, ce phénomène n'apparaît que lorsque les fréquences en jeu sont suffisamment proches.

Mécanisme de sélection fréquentielle

Lorsque la platine de l'étrier met en mouvement la fenêtre ovale, elle impose une différence de pression dans la périlymphe, donc entre les deux compartiments cochléaires². Cette pression différentielle met en vibration la membrane basilaire (MB) et produit une déformation dont le maximum se situe à une position qui dépend de la fréquence du stimulus (lieu tonotopique). En régime linéaire, la pression différentielle $\bar{P} = P_2 - P_1$ se propage le long de la cochlée sous la forme d'une onde à faible vitesse de phase et obéit à l'équation $(\partial_x^2 + k^2) \bar{P} = 0$, où k est un nombre d'onde complexe. Les courbes de la Figure 3.13 C montrent la partie réelle (orange, propagation) et la partie imaginaire (bleue, amortissement) de k en fonction de la fréquence normalisée f/f_c , ce qui explique la focalisation de l'énergie autour de la fréquence caractéristique [102].

La mise en mouvement de la membrane basilaire entraîne des perturbations dans le fluide du canal cochléaire (l'endolymphe). Ces perturbations induisent un déplacement latéral des cellules ciliées internes (CCI), qui viennent alors en contact avec la membrane tectoriale. Ce contact provoque la flexion des stéréocils, modulant l'ouverture de leurs canaux potassiques et déclenchant ainsi la stimulation des fibres nerveuses afférentes.

La structure de ces stéréocils (masse, longueur et rigidité) diffère le long de la cochlée, ce qui produit des variations de leurs propriétés mécanique. Combinées au couplage lieu-fréquence de la déformation de la membrane basilaire, ces variations constituent les éléments clés du mécanisme de sélection fréquentielle. De ce fait, les régions à la base de la cochlée sont particulièrement sensibles aux hautes fréquences, tandis que celles situées à l'apex répondent préférentiellement aux basses fréquences.

Néanmoins, la compréhension de ce mécanisme est longtemps restée incomplète. Avant les années 1980, les expériences sur la cochlée étaient réalisées *post mortem*, jusqu'à ce que l'on découvre que son comportement différait sensiblement *in vivo*. Ces mesures, représentées en Figure 3.13 A, indiquent qu'il faut environ 50 dB supplémentaires pour obtenir une vibration équivalente. Cela montre que la modulation de la membrane basilaire dépend non seulement de la fréquence et de la zone analysée, mais aussi d'un apport actif d'énergie. Nous verrons que cet apport est fourni par l'électromotilité des cellules ciliées externes (CCE) : un mécanisme modulé par les voies efférentes du tronc cérébral. Sans ce mécanisme actif, la discrimination fine des sons serait impossible.

2. Les canaux (ou rampes) tympanique et vestibulaire.

D'une manière générale, plus une fréquence s'éloigne de son point de résonance, plus l'intensité du stimulus doit être élevée pour induire une vibration détectable de la membrane. Comme l'illustre la figure 3.13 B, en dehors de sa zone caractéristique, la membrane basilaire tend vers un régime de réponse linéaire vis-à-vis du niveau sonore³.

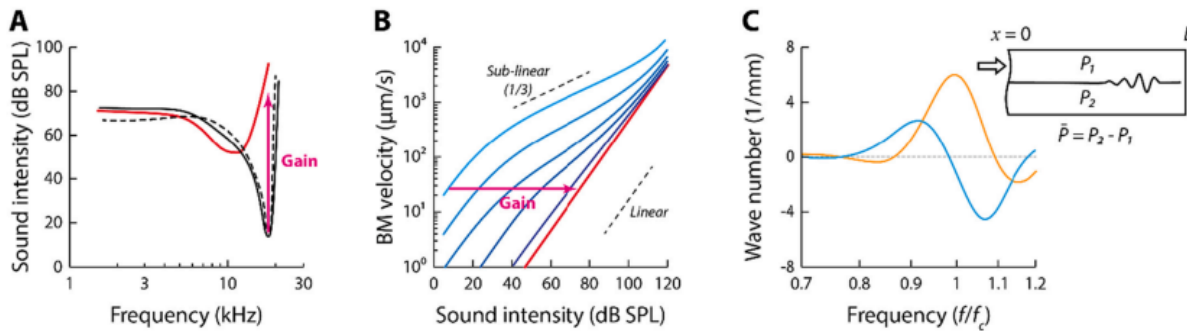


FIGURE 3.13 – Caractérisation de la mécanique cochléaire.

A. Niveau d'intensité sonore requis, à différentes fréquences, pour atteindre le seuil minimal de mouvement de la membrane basilaire. Analyse située à la base de la cochlée. Ligne noire continue : analyse sur une cochlée in-vivo ; ligne rouge : analyse sur une cochlée post-mortem. Données d'après [103, 104]. **B.** Relation entre la vitesse de la membrane basilaire et le niveau sonore du stimulus. En cyan, évolution lorsque la fréquence correspond à la fréquence de résonance locale de la membrane. Bleu foncé, évolution pour des fréquences plus élevée. La ligne rouge correspond à des mesures post-mortem. Les échelles linéaires et sublinéaires (1/3) sont indiquées à titre indicatif. Valeurs d'après [104]. **C.** Représentation du nombre d'onde k en fonction de f/f_c . En orange la partie réelle, en bleue la partie imaginaire. D'après les travaux de [102].

Figure issue des travaux de [91].

Les cellules ciliées internes (CCI)

Comme expliqué précédemment, les CCI convertissent les vibrations mécaniques en signaux nerveux. On en dénombre environ 3 500, disposées en une seule rangée le long de la cochlée humaine, contre trois rangées de CCE. Bien qu'en nombre inférieur, les CCI reçoivent la majeure partie de l'innervation afférente : environ 90–95 % des fibres auditives (type I) leur sont dédiées (flèche noire sur la Fig. 3.16 A). En moyenne, chaque CCI est connectée par 10–20 fibres du ganglion spiral.

A leur extrémité sont rangés des stéréocils de taille croissante. Chacun de ces stéréocils est relié à son voisin le plus grand par une chaîne de protéine (cadherin-23/protocadherin-15, aussi appelées *Tip links*). Contrairement aux CCE, les stéréocils des CCI ne sont pas reliés à la membrane tectoriale et sont pliés uniquement par le mouvement de l'endolymphe. Lorsqu'un cil se plie, les chaînes de protéine tirent ou poussent sur le stéréocil voisin et ouvrent ou referment respectivement un canal de mécanotransduction (MET).

Lorsque ce canal est ouvert, des ions (principalement du potassium : K⁺) polarisent la cellule interne et provoquent l'ouverture de canaux calcique voltage-dépendant (Ca_v1.3).

3. En dB, donc logarithmique par rapport à la pression.

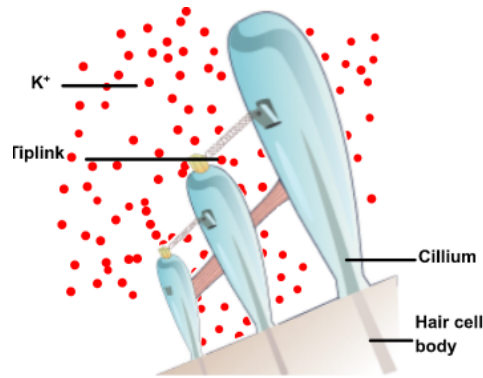


FIGURE 3.14 – Transduction mécanique des cellules ciliées. L'ouverture de ce canal s'agrandit ou se rétrécit en fonction du sens de repliement du cil.

Image source : <https://nba.uth.tmc.edu/neuroscience/m/s2/chapter12.html>

Cette entrée d'ions-calcium (Ca^{2+}) déclenche l'exocytose de vésicules de *glutamate*, un neurotransmetteur à la base de la cellule, qui va stimuler des fibres nerveuses en rubans. Ces synapses en ruban sont idéalement adaptées pour garantir des taux d'émissions soutenus et produire une réponse synaptique graduée plutôt qu'une réponse « tout ou rien » comme on le trouve chez les neurones à décharge (spiking neurones). Dans les CCI, chaque fibre afférente établit une seule synapse avec un ruban ⁴ [105, 106].

N. Marin et al.

Molecular and Cellular Neuroscience 120 (2022) 103732

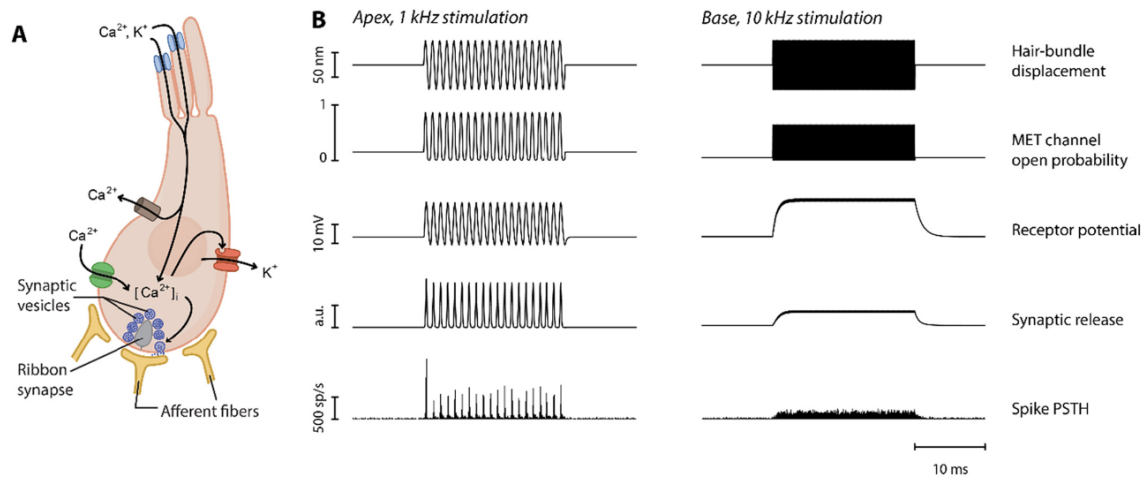


FIGURE 3.15 – Encodage sonore dans le nerf auditif.

Figure issue de [91].

Le potentiel récepteur suit une relation sigmoïde entre la déflexion du faisceau de stéréocils et la probabilité d'ouverture des canaux MET. Au voisinage de $P_o \approx 0.5$ (Figure 3.16 B), la pente de cette sigmoïde est maximale, optimisant la sensibilité aux déflexions lentes. En revanche, pour de faibles stimuli sinusoïdaux à haute fréquence, la réponse

4. Vrai chez la souris, mais il peut exister des variation en fonction de l'espèce et de la position cochléaire.

électrique est limitée par le filtrage passe-bas imposé par la constante de temps membranaire ($\tau = R_m C_m$) : la composante alternative (AC) du potentiel transducteur des CCI décroît rapidement et, au-delà de quelques kHz [107], devient surtout continue (DC).

Comme l'illustre la Figure 3.15, pour différencier les modulation de réception en haute fréquence (HF) et éviter les erreurs de stimulus ponctuels, le cerveau analyse dans ce cas plutôt l'augmentation du courant MET moyen (*train de spike*) que le signal et cela, sur un domaine fréquentiel (positions dans la cochlée) de plus en plus large. Cela explique pourquoi notre perception suit une échelle logarithmique en HF, et pourquoi une concentration plus dense de cellules dans cette zone n'aurait pas d'intérêt à cause des limites biochimiques de notre système nerveux.

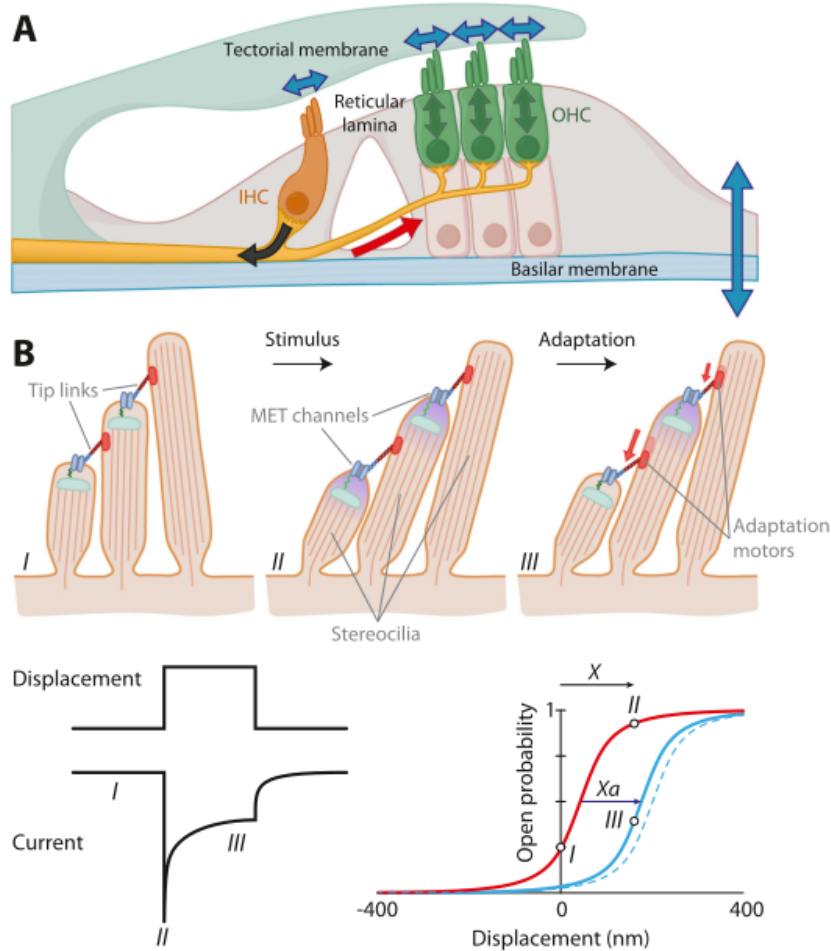


FIGURE 3.16 – Schema représentant l'adaptation des stéréocils.
Figure issue de [91]

Pour optimiser cette détection « en DC », la probabilité d'ouverture de ces canaux est modulée par un phénomène d'adaptation (Fig. 3.16 B, situation III), ce qui permet de maintenir une détection fine de la fréquence même lorsque la stimulation est forte. Dans ce type de situation, le point de repos du faisceau est abaissé autour de $P_o \approx 0.2$. À cet endroit, la courbure de la relation sigmoïde entre déflexion et probabilité d'ouverture est maximale, de sorte qu'une petite oscillation sinusoïdale génère un décalage DC mesurable

(rectification) même si la composante AC est filtrée. Ce processus permet aux cellules ciliées de continuer à répondre avec une sensibilité élevée alors que la réponse satureraient autrement.

Ce principe éclaire l'observation, en apparence contre-intuitive, selon laquelle les CCI à basse fréquence présentent des réponses plus rapides que celles dédiées aux fréquences élevées, ces dernières codant préférentiellement l'enveloppe du stimulus et non le signal lui même [108].

Fait notable, l'architecture de base demeure très similaire chez les mammifères écholocalisateurs (p. ex. les chauves-souris). Si les limites de la constante de temps membranaire des CCI réduisent la possibilité de synchronisation temporelle à haute fréquence, l'audition ultrasonore s'appuie sur des adaptations complémentaires : une membrane basilaire basale plus étroite et rigide (parfois avec « fovéa acoustique »), une électromotilité des CCE optimisée par des adaptations de la protéine prestin, et des circuits centraux spécialisés (CF-CF, FM-FM) pour le suivi des décalages Doppler et de l'enveloppe, en complément de la discrimination spectrale classique.

Les cellules ciliées externes (CCE)

Il y a environ 12 000 cellules ciliées externes, disposées en trois rangées le long de la cochlée humaine et reliées à la membrane tectoriale.

En vibrant, la membrane basilaire réduit sa distance avec la membrane tectoriale, ce qui fait fléchir les stéréocils des CCE reliés de part et d'autre et entraîne une modulation du potentiel de ces cellules. Dans le cas d'un son pur, les cils vont osciller de manière sinusoïdale ce qui va induire une succession de dépolarisation et de repolarisation.

Contrairement à ce qui se passe dans les CCI, ces variations de potentiels provoquent un changement de la forme des CCE. Quand elles sont dites hyperpolarisées (rejet d'ion K+) elles raccourcissent, lorsqu'elles se repolarisent, elles s'allongent. Ce phénomène biomécanique très particulier, nommé **l'électro-motilité**, est due à un changement de forme d'une protéine très abondante dans la membrane cellulaire : *la prestine*.

En s'appuyant sur la membrane tectoriale et en changeant en permanence de longueur, les CCE jouent le rôle d'oscillateurs qui amplifient très localement la vibration de la membrane basilaire (tonotopie active). Cette propriété des CCE augmente la sensibilité de la cochlée d'environ 50-60 dB et améliore sa capacité à coder de manière différentielle des fréquences très proches. Sans cette activité, il serait quasiment impossible de distinguer les sons [109].

A la différence des CCI, les connexions des CCE avec le cerveau sont principalement efférentes (flèche rouge Fig. 3.16 A). Ces neurones peuvent libérer de manière volontaire une molécule : l'acétylcholine, qui va forcer la dépolarisation des CCE. Autrement dit, le cerveau a la capacité de réguler la sensibilité de ces cellules, modulant ainsi leurs amplifications et permettant d'affiner ou non la détection d'une bande sonore perçue

par les CCI. Ce mécanisme permet d'améliorer dynamiquement l'attention auditive, mais aussi de réduire la modulation de la membrane en cas de bruit trop fort (pour protéger les CCI).

Cette activité produit des otoémissions acoustiques (OEA) qui persistent brièvement après la stimulation et peuvent être enregistrées expérimentalement à l'aide d'un microphone placé dans le conduit auditif. Les OEA peuvent être spontanées, transitoires ou par produits de distorsion, et fournissent des informations précieuses sur l'état de l'oreille interne. En clinique, la présence de ces otoémissions est le signe d'un bon fonctionnement des CCE ; leur absence indique au contraire que ces cellules sont endommagées et peut expliquer des cas de surdité.

Questions encore ouvertes

Il est admis que l'électromotilité des CCE constitue un élément essentiel du processus d'amplification cochléaire chez les mammifères. Toutefois, plusieurs points empêchent de clore le débat.

Tout d'abord, l'électromotilité est commandée par le potentiel transmembranaire, et non par le courant récepteur [110]. Le modèle électrique d'une bicouche lipidique portant des canaux se résume à une capacité en parallèle avec une résistance, formant un filtre passe-bas dont la constante de temps atteint typiquement quelques millisecondes dans la plupart des cellules. Même avec les propriétés électriques rapides des CCE, la composante variable du potentiel serait fortement atténuée au-delà d'environ 1 kHz [111], ce qui rendrait ce mécanisme inopérant aux hautes fréquences.

Des mesures tomographiques de vibrations ont conduit à des conclusions divergentes. D'une part, des vibrations fortement *rectifiées* ont montré que les CCE se comportent comme des filtres passe-bas du premier ordre, avec des fréquences de coupure proches de 3 kHz (soit plus de 2.5 octaves en dessous des fréquences caractéristiques correspondantes) suggérant un rôle de détecteurs d'enveloppe [112]. D'autre part, une analyse fine des déplacements verticaux des CCE a mis en évidence des mouvements de directions opposées à des fréquences dépassant 20 kHz, compatibles avec des variations rapides de longueur somatique [113]. Pour contourner la limitation passe-bas, il a été proposé que l'électromotilité ne suive pas le potentiel transmembranaire, mais plutôt le potentiel extracellulaire non filtré [114]. Des travaux plus récents suggèrent par ailleurs que le potentiel de repos des CCE est suffisamment dépolarisé (environ ~ 50 mV) pour activer une conductance potassique dépendante du voltage (K^+), raccourcissant la constante de temps membranaire à $\approx 25 \mu s$, ce qui correspond à une fréquence de coupure d'environ 6.4 kHz [115].

Néanmoins, bien qu'une telle constante de temps élargisse la bande passante des mouvements électromotiles, la fréquence de coupure associée demeure inférieure à la fréquence caractéristique d'environ 10 kHz de la CCE correspondante. De même, la manière dont l'électromotilité pourrait fonctionner au voisinage de la limite supérieure de l'audition chez certaines chauves-souris (~ 100 kHz) reste une question ouverte.

Dans cette perspective, les CCE ne se limiteraient pas à un rôle d’amplification mécanique : leurs synapses afférentes, bien que peu nombreuses, pourraient également participer à des fonctions de signalisation spécifiques. Ces voies neuronales seraient sollicitées à fortes intensités sonores, contribuant potentiellement à des mécanismes de protection cochléaire [116]. Toutefois, des travaux récents suggèrent que ces circuits pourraient aussi intervenir à des niveaux d’intensité modérés, laissant entrevoir une implication plus large des CCE dans la perception acoustique elle-même.

En conclusion, plusieurs incertitudes demeurent quant à la compréhension complète de ce système. Les corrélations entre les observations biophysiques (telles que la structure moléculaire des CCE, leurs connexions neuronales et leur dynamique électromécanique) et les performances auditives observées chez les mammifères ne sont pas encore entièrement élucidées. La seule chose totalement admise à ce jour est que les CCI assurent la transduction et l’analyse fréquentielle des signaux acoustiques, tandis que les CCE participent activement à l’amplification et au raffinement de la sensibilité auditive, sous l’influence des voies efférentes et de mécanismes attentionnels.

3.5 Liaisons neuronales oreille-cerveau

Encodage du signal dans le ganglion spiral

Gradients tonotopiques et propriétés intrinsèques

Les propriétés intrinsèques des neurones du ganglion spiral varient selon la position tonotopique. En culture, les neurones connectés à la base cochléaire présentent des latences plus courtes, des constantes de temps membranaires réduites et, paradoxalement, des taux de décharge évoqués plus faibles [117, 118].

D’autres paramètres suivent une distribution en cloche le long de la cochlée : les seuils de décharge sont minimaux pour les fréquences moyennes, c’est-à-dire là où la sélectivité fréquentielle est la plus élevée. Cette région est également innervée par des neurones de plus petite taille, possiblement en lien avec leurs propriétés membranaires et l’organisation contrainte du modiolus (axe osseux spongieux de la cochlée).

Codage temporel (verrouillage de phase).

La libération synaptique au niveau des CCI déclenche une dépolarisation suffisante des fibres afférentes de type I (neurones du ganglion spiral, SGN) pour générer des potentiels d’action. Lorsque le stimulus est périodique, les potentiels surviennent préférentiellement en phase avec ce stimulus (*phase locking*). On admet aujourd’hui que ce codage temporel est robuste jusqu’à ~ 2 kHz chez la plupart des mammifères [119, 120]. Au-delà, des limitations biophysiques empêchent le verrouillage de phase, et le codage temporel devient essentiellement *régional* (basé sur la position du neurone). Dans ces cas, l’information est portée par l’enveloppe du signal (modulations lentes) et requiert une intégration spatiale sur des plages cochléaires plus étendues pour rester robuste, leur largeur augmentant exceptionnellement avec la fréquence.

Codage d'intensité (courbes taux–niveau).

À faibles niveaux (et pour des fréquences où le verrouillage de phase est efficace), la probabilité instantanée de décharge croît de manière quasi exponentielle avec le niveau de pression acoustique [121]. Ce constat motive l'usage des décibels (dB), une échelle logarithmique, pour quantifier le niveau sonore.

Lorsque le niveau augmente vers des valeurs plus élevées, on observe une saturation progressive à différents maillons de la chaîne (courant de transduction, libération synaptique, génération des potentiels d'action), de sorte que le taux moyen de décharge des neurones sensoriels de type I présente une relation de type sigmoïde avec l'intensité sonore [120]. La large gamme dynamique auditive est alors couverte par une *population* hétérogène de fibres (aux seuils et taux spontanés différents), chacune offrant une pente optimale sur une portion de niveaux, tandis que l'ensemble fournit un code d'intensité robuste.

Notons enfin que l'adaptation rapide (de l'ordre de la dizaine de millisecondes) façonne une réponse *phasique-tonique* : un pic d'onset suivi d'un palier, ce qui améliore la détection des débuts et des fins de sons tout en limitant la saturation soutenue.

Codage spatial

Le complexe olivaire supérieur joue un rôle central dans la localisation sonore (azimut et élévation de la source). L'olive médiale supérieure (MSO) est spécialisée dans l'estimation des différences interaurales de temps (*interaural time differences*, ITD), un indice particulièrement informatif aux basses fréquences, tandis que l'olive latérale supérieure (LSO) code préférentiellement les différences interaurales de niveau (*interaural level differences*, ILD), plus saillantes aux hautes fréquences [122, 123].

Ce « duplex coding » n'est toutefois pas exclusif : il existe des chevauchements de sensibilités et des contributions d'autres relais, mais MSO et LSO constituent les premiers sites binauraux majeurs du tronc cérébral chez les mammifères.

Attention auditive

L'attention auditive se décline en deux formes complémentaires : une composante *bottom-up*, déclenchée par la saillance des événements sensoriels dans l'environnement, et une composante *top-down*, guidée par une volonté propre de l'individu (conscience) et relatif à une tâche souhaitée [124].

Cette sélection peut se comprendre comme un *filtrage* aligné sur l'analyse fréquentielle issue de la cochlée : l'auditeur « calibre » des filtres attentionnels (en fréquence, dans le temps et en modulation spectro-temporelle) pour privilégier les caractéristiques pertinentes du signal et réprimer les distracteurs [125].

Au niveau périphérique, le contrôle *top-down* s'exerce via le *faisceau olivocochléaire* (OCB). La voie médiane (MOC), qui part du tronc cérébral médial et innerve directement les

CCE⁵, peut réduire le gain de l'amplificateur cochléaire, et ainsi, moduler la sélectivité fréquentielle transmise aux CCI. La voie latérale (LOC), qui part du complexe olivaire latéral vers les dendrites des fibres afférentes de type I⁶, semble quand à elle surtout ajuster l'excitabilité (*set-point*) des CCI, rôle pressenti (mais encore débattu) dans la protection aux bruits forts et pour une attention sélective supplémentaires du codage [126,127].

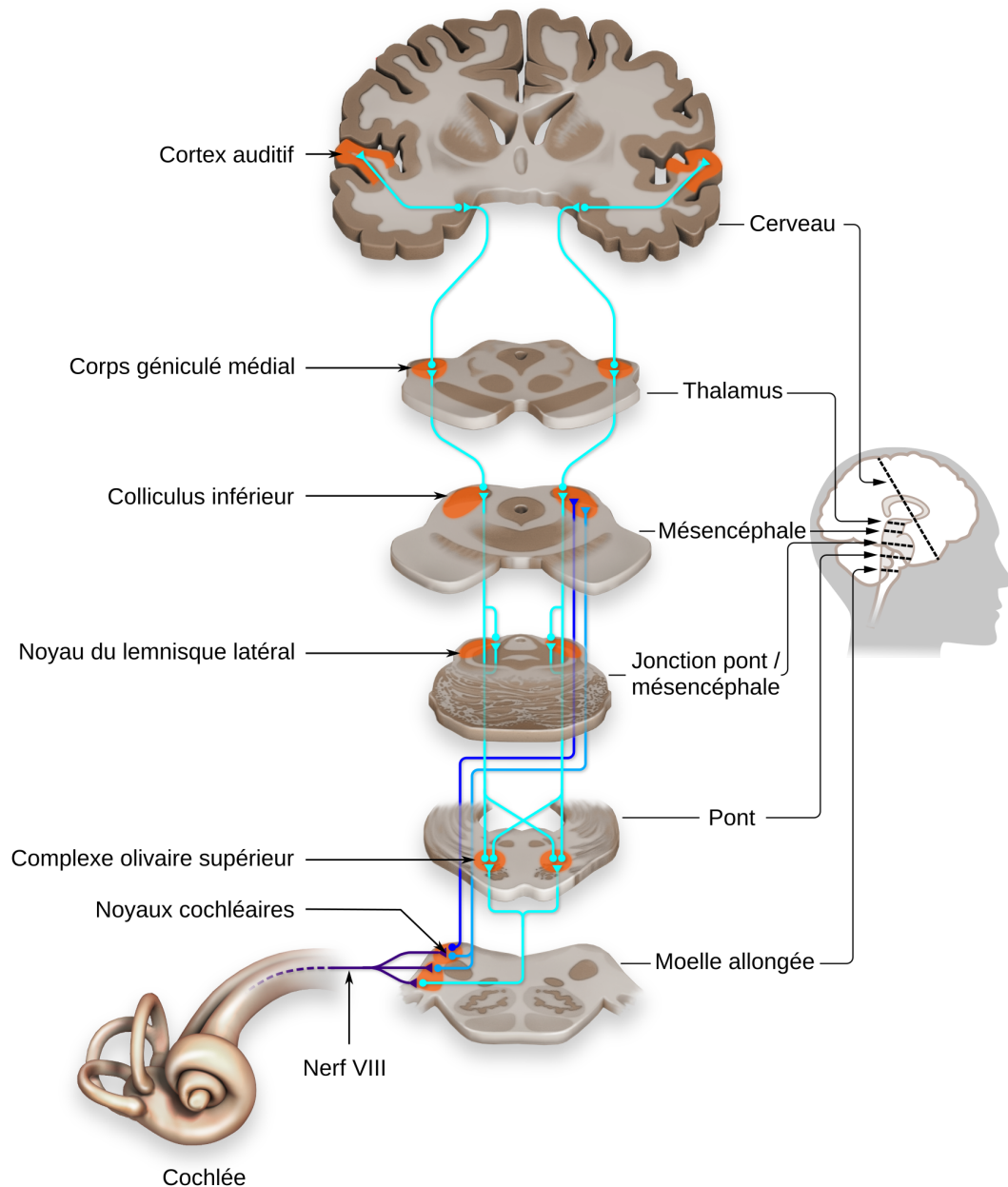


FIGURE 3.17 – Illustration simplifiée de la voie auditive ascendante (bleu clair) et descendante (bleu foncé). À chaque étape, une partie des signaux est relayée ou traitée localement, tandis qu'une autre poursuit sa route vers les structures supérieures. (Par Emily Cooper et sous licence CC BY 4.0)

5. Ces fibres sont *myélinisées*, ce qui provoque une réponse rapide.

6. Située en dessous des CCI. Ces fibres sont *non myélinisées* (action plus tonique).

En parallèle, des mécanismes neurophysiologiques plus centraux agissent comme des filtres spectro-temporels dynamiques qui alignent la sensibilité neuronale sur les patrons du son cible [128]. Plus récemment, des enregistrements cochléaires directs chez des porteurs d'implant ont mis en évidence une modulation anticipatoire (thêta) de l'activité du nerf auditif par l'attention [129].

Ces interactions multi-niveaux (périphérie–tronc cérébral–cortex) s'articulent avec l'organisation tonotopique et les « filtres » cochléaires, comme synthétisé dans la Fig. 3.17 [130].

Comparaison numérique

Au niveau cortical, des travaux comparant des *front-ends* cochléaires biophysiques détaillés à des représentations plus simples de type spectrogramme montrent que, pour prédire les réponses unitaires à des sons naturels, des modèles « simples » (banque de filtres log-spacés avec compression quasi logarithmique) atteignent des performances similaires aux modèles biophysiques les plus sophistiqués [131]. Cela suggère qu'une large part du décodage cortical s'appuie sur des informations spectro-temporelles déjà disponibles via la tonotopie et le traitement précoce.

Partie II

Contributions

Contribution 1 : Une nouvelle métrique d'évaluation - le PPI

4.1 Motivations

Cette thèse propose un ensemble de méthodes et de solutions matérielles visant à concevoir des intelligences artificielles nécessitant des puissances proches de l'ultra basse consommation ($\approx 250 \mu\text{W}$). Pour contextualiser cet objectif au regard des technologies actuelles, les tableaux 4.1 et 4.2 présentent respectivement des estimations de l'autonomie de différentes batteries en fonction de la puissance consommée, ainsi que des exemples représentatifs de dispositifs d'IA pour différents niveaux de consommation énergétique¹.

TABLE 4.1 – Autonomie estimée en fonction de la puissance consommée et du type de batterie. *valeurs arrondies selon l'ordre de grandeur pertinent à des fins de lisibilité.*

Batterie Puissance requise	Pile bouton CR2032 (0.705 Wh)	Pile AA Lithium (4.5 Wh)	iPhone 13 Batt. (12.16 Wh)	Powerbank 20Ah (74 Wh)
100 μW	10 mois	5 ans	14 ans	84 ans
1 mW	29 jours	6 mois	17 mois	8 ans
10 mW	3 jours	19 jours	2 mois	10 mois
100 mW	7 h	2 jours	5 jours	1 mois
1 W	42 min	4 h 30 min	12 h	3 jours
10 W	N/A	N/A	N/A	7 h
100 W	N/A	N/A	N/A	42 min

Une tendance bien établie en intelligence artificielle montre que plus une tâche est complexe, plus le coût calculatoire nécessaire au fonctionnement du réseau de neurones est élevé. Par ailleurs, plusieurs travaux ont mis en évidence une relation entre la puissance de calcul (en opérations flottante par seconde) fournie par les architectures conventionnelles et leur consommation énergétique.

1. Remarque : les valeurs indiquées correspondent à des estimations idéales ne tenant pas compte des pertes dues à l'auto-décharge, souvent significatives pour les batteries Li-Po mais négligeables pour les piles lithium-thionyle.

TABLE 4.2 – Exemples de matériels d’IA ASIC classés par puissance consommée, avec une tâche illustrant les performances d’inférence.

Puissance	Matériel IA	Tâche / Performance (approx.)
100 μ W	Syntiant NDP102 – 1.71mm x 2.51mm 20-pin eWLB	Détection continue (mot-clé ou anomalie)
1 mW	Syntiant NDP200 – puce vision ultra-basse consommation	Détection d’objet basique (1 inf/s, 1 mJ/inf)
10 mW	Maxim MAX78000 – microcontrôleur IA	Reconnaissance vocale en continu (1 inf/s, 8 mJ/inf)
100 mW	Espressif ESP32 – SoC IoT + DSP	Analyse embarquée capteurs / vision simple (20 inf/s, 5 mJ/inf)
1 W	Google Coral Edge TPU – ASIC d’inférence	Classification d’images rapide (400 inf/s, 3,4 mJ/inf)
10 W	NVIDIA Jetson Nano – SoC CPU+GPU	Vision temps réel (36 inf/s sur ResNet-50, 280 mJ/inf)
100 W	Habana Labs Goya HL-1000 – ASIC datacenter	Classification d’images à haute cadence (15 000 inf/s, 6,7 mJ/inf)

La figure 4.1, extraite du manuscrit HDR de V. Gies [132], illustre cette tendance. Elle met en évidence une relation empirique entre la puissance de calcul C (en FLOPS)² et la puissance énergétique P (en Watts), décrite par l’équation suivante :

$$\log_{10}(C) = 1.17 \cdot \log_{10}(P) + 9.7 \quad (4.1)$$

Cette thèse ambitionne de dépasser cette tendance de deux ordres de grandeur (ligne bleue) grâce à une rupture architecturale, détaillée dans le chapitre 5. Toutefois, la contrainte d’ultra-basse consommation implique une réduction significative de la puissance de calcul disponible. En comparaison avec des architectures d’IA très énergivore, cette limitation peut se traduire par une dégradation des performances applicatives.

Dès lors, une question essentielle se pose : comment évaluer de manière objective le compromis entre performance et consommation énergétique pour différents modèles d’IA ?

Pour répondre à cette question, nous proposons une extension de l’analyse précédente, avec la définition d’un nouvel indice d’évaluation : le **Performance-Power Index (PPI)**. Cette première contribution vise à fournir un score, applicable à différentes tâches d’usage, permettant de quantifier la performance d’un modèle relativement à son efficacité énergétique.

2. Attention, il s’agit du nombre d’opérations flottantes par seconde (FLOPS) pour les grandes valeurs et du nombre d’instructions par seconde (IPS) pour les petites valeurs. Dans des microcontrôleurs, il est fréquent qu’une opération flottante s’exécute en plusieurs instructions (≈ 3 avec un FPU), tandis que sur des CPU ou GPU performants, cela se fait en une seule instruction.

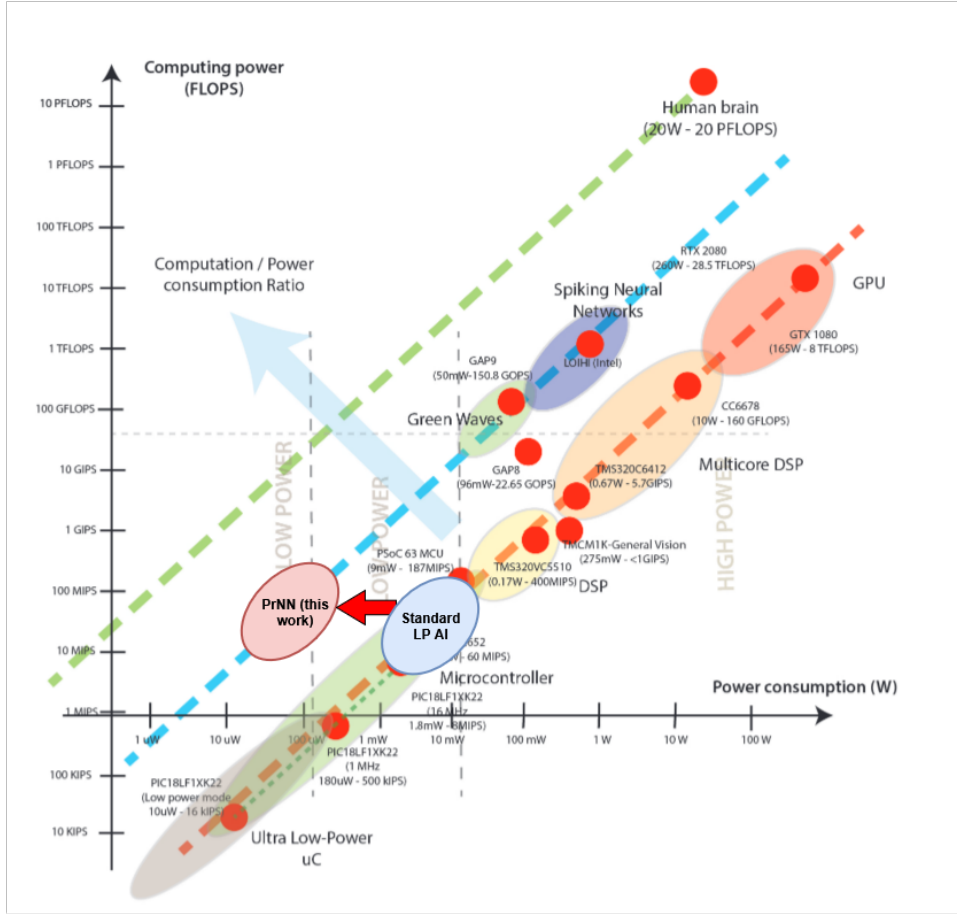


FIGURE 4.1 – Relation entre la puissance de calcul et la consommation énergétique pour différentes architectures numériques. Figure issue de l'HDR de V. Gies (2021).

4.2 Formulation mathématique

Méthodologie de conception

La métrique que nous souhaitons développer dépendra de deux paramètres : la **performance** finale du modèle, représentée par l'aire sous la courbe précision-rappel (*Precision-Recall Area Under Curve*, notée PR-AUC) et l'**énergie totale** dépensée en moyenne par l'IA (en Watts)³.

Pour que notre indice soit robuste, interprétable et comparable, nous imposons un certain nombre de contraintes à la fonction recherchée.

Tout d'abord, nous souhaitons que cette fonction soit définie et continue sur l'ensemble $[0, 1] \times \mathbb{R}$ (domaine des paramètres), de classe C^1 , et à valeurs dans $\mathbb{R}^+$. Elle doit être strictement décroissante lorsque la consommation énergétique augmente ou que la performance diminue. De manière logique, une IA dont la Pr-AUC est inférieure ou égale à

3. L'estimation de l'énergie requise par le réseau de neurones numérique pourra se faire grâce à l'équation 4.1

0.5 (performance aléatoire) doit obtenir un score nul, quelle que soit sa consommation énergétique.

Par ailleurs, il faut garder à l'esprit que notre objectif est de comparer aussi bien des modèles très énergivores sur des tâches complexes que des modèles sobres sur des tâches plus simples. Pour assurer cette pertinence, la croissance de la fonction doit suivre certaines tendances : lorsque la puissance requise varie, le score doit évoluer de manière linéaire sur une échelle logarithmique, car c'est l'ordre de grandeur qui nous intéresse ; en revanche, lorsque la PR-AUC varie, le score doit évoluer de manière exponentielle, car en pratique l'effort nécessaire pour améliorer la performance croît exponentiellement avec le logarithme de la complexité opératoire. Cependant, au-delà d'une PR-AUC de 0.9, nous considérons que le jeu de données d'évaluation atteint une forme de saturation et que sa capacité à discriminer entre les modèles devient discutable. Afin de limiter l'impact de variations aléatoires mineures entre deux modèles déjà performants, nous souhaitons une croissance logarithmique au-delà de ce seuil, de manière à stabiliser le PPI et d'éviter qu'il ne reflète des différences non significatives.

En tenant compte de l'ensemble de ces contraintes, nous obtenons une représentation graphique du PPI illustrée dans la figure 4.2, et dont la formule générale est donnée par :

$$\text{PPI} = \gamma_{\text{regul}} (f_{\text{perf}}(\text{PR-AUC}) + f_{\text{power}}(P), \text{PR-AUC}) \quad (4.2)$$

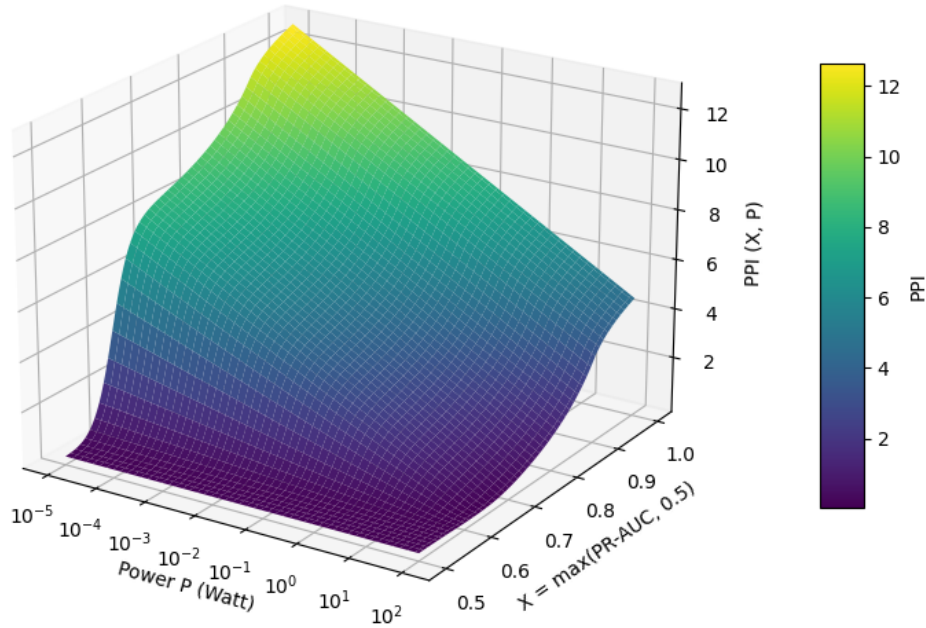


FIGURE 4.2 – Représentation graphique du Performance-Power Index (PPI) en fonction du score de PR-AUC et de la puissance requise (en Watt).

Construction des fonctions f_{perf} et f_{power}

Nous introduisons d'abord la variable

$$X = \max(\text{PR-AUC}, 0.5),$$

de manière à ne considérer que la portion de la PR-AUC reflétant effectivement la capacité discriminante du modèle. La fonction f_{perf} est alors définie comme suit :

$$f_{\text{perf}}(X) = \begin{cases} (-\log_2(1 - X) - 1)^k, & \text{si } X \leq x_0, \\ (-\log_2(1 - x_0) - 1)^k + A \cdot \ln\left(1 + B \cdot \frac{X - x_0}{1 - x_0}\right), & \text{si } X > x_0. \end{cases} \quad (4.3)$$

Les paramètres sont fixés de manière arbitraire à $k = 1.5$, $x_0 = 0.9$, $A = 1.006$ et $B = 3.279$, de sorte que l'on ait $f_{\text{perf}}(1) \approx 5$.

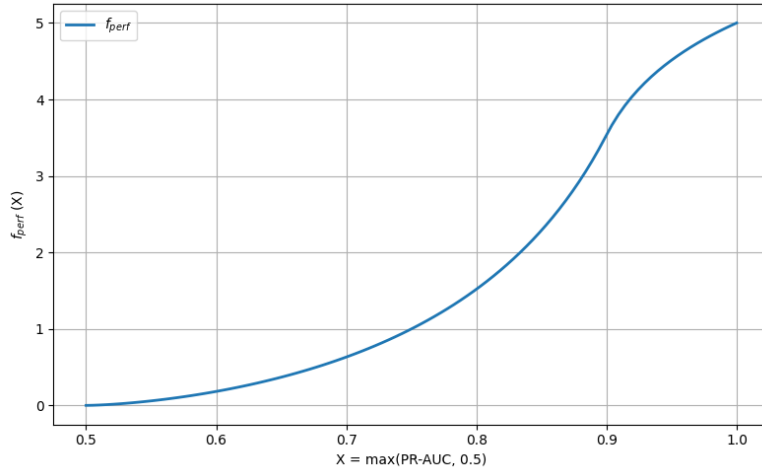


FIGURE 4.3 – Représentation graphique de la fonction $f_{\text{perf}}(X)$.

La fonction f_{power} est définie plus simplement par :

$$f_{\text{power}}(P) = -1.17 \cdot \log_{10}(P) + 1.9. \quad (4.4)$$

Construction de γ_{regul}

Mathématiquement, il est important que les fonctions f_{perf} et f_{power} interviennent séparément dans la formule du PPI⁴ pour que le score soit représentatif quelle que soit la complexité de la tâche considérée. Cependant, cette sommation pose problème dans certains cas extrêmes : par exemple, un modèle aléatoire ($f_{\text{perf}} = 0$) consommant très peu de ressources ($f_{\text{power}} \rightarrow +\infty$) pourrait théoriquement obtenir un PPI supérieur à celui d'un modèle performant ($f_{\text{perf}} \approx 5$) mais plus énergivore, ce qui serait incohérent.

4. Autrement dit, sous forme additive et non une multiplication.

Pour éviter cette situation, nous introduisons une fonction de régularisation γ_{regul} , définie par :

$$\gamma_{\text{regul}}(f_{\text{perf}} + f_{\text{computing}}, X) = \frac{1}{\beta} \ln \left(1 + e^{\beta \cdot \sigma_{\text{regul}}(X) \cdot (f_{\text{perf}} + f_{\text{computing}})} \right). \quad (4.5)$$

La fonction σ_{regul} agit ici comme un facteur d'écrasement dépendant de la qualité du modèle :

$$\sigma_{\text{regul}}(X) = \frac{1}{1 + \exp(-50 \cdot (\max(\text{PR-AUC}, 0.5) - 0.6))}. \quad (4.6)$$

Cette fonction logistique a pour effet de ramener le PPI vers zéro lorsque la PR-AUC est insuffisante pour garantir une utilité pratique. En pratique, nous considérons qu'un modèle ayant une PR-AUC inférieure à 0.6 échoue dans sa tâche, et doit donc être fortement pénalisé, indépendamment de sa consommation énergétique. Le reste de la formulation de γ_{regul} reprend la fonction *Softplus*, qui assure une sortie strictement positive ($\mathbb{R}^+$) même lorsque la somme $f_{\text{perf}} + f_{\text{computing}}$ est négative, tout en garantissant la continuité et la dérivabilité sur l'ensemble de son domaine. Nous fixons ici $\beta = 10$.

En résumé, la formulation développée du PPI est :

$$\text{PPI}(X, P) = \frac{1}{10} \ln \left(1 + e^{\frac{10 \cdot (f_{\text{perf}}(X) + f_{\text{power}}(P))}{1 + e^{-50(X - 0.6)}}} \right). \quad (4.7)$$

Contribution 2 : Architecture des Primitive Neural Networks

5.1 Les limites du paradigme actuel

Bien que l'intelligence artificielle ait, ces dernières années, démontré des performances comparables, voire supérieures, à celles de l'être humain dans de nombreux domaines, un écart significatif demeure en matière de consommation énergétique. En nous appuyant sur notre contribution précédente, nous constatons que les meilleurs indices Performance-Puissance (PPI) des architectures d'IA traditionnelles restent inférieurs de près de 4 points à ceux atteints par leurs équivalents biologiques. Cet écart est particulièrement marqué dans les tâches exécutées à haute fréquence, telles que la reconnaissance visuelle ou auditive en temps réel, et ce malgré l'utilisation de méthodes d'optimisation énergétique telles que celles décrites dans le chapitre 2.

Cette limitation découle en réalité du paradigme numérique adopté, qui contraint à concentrer les efforts d'optimisation exclusivement sur la consommation énergétique du réseau de neurones, alors que d'autres étapes interviennent également dans la chaîne de traitement. Les plus notables étant : l'échantillonnage à haute fréquence du signal, sa mise en mémoire tampon et les prétraitements qui lui sont appliqués (comme la transformée de Fourier dans le cas de l'analyse acoustique). Ces opérations introduisent des surcoûts énergétiques fixes qui peuvent être significatifs pour des systèmes visant une faible consommation.

À l'inverse, l'évolution des systèmes biologiques a conduit à une optimisation conjointe de l'ensemble de la chaîne perceptive. Non seulement la structure du centre de traitement neuronal (le cerveau) a été affinée, mais les organes sensoriels qui y sont connectés pour prétraiter les signaux ont également été optimisés. Ce processus de co-évolution permet aux systèmes biologiques de limiter la charge cognitive tout en maintenant une grande sensibilité contextuelle. Il en résulte une efficacité énergétique nettement supérieure à celle des architectures numériques actuelles.

Certains travaux récents ont commencé à répondre à cette problématique en adoptant une démarche de co-conception entre les capteurs matériels et les modèles neuronaux associés. Par exemple, SACoD [133] optimise conjointement un capteur d'imagerie à codage de phase et un réseau de neurones convolutif différentiable, afin de réduire le coût énergétique global de l'inférence. De manière similaire, des systèmes tels que BlissCam [134] implémentent un échantillonnage parcellaire des pixels directement au niveau du capteur, réduisant ainsi la bande passante d'entrée sans nécessairement compromettre la qualité des performances. Enfin, les caméras événementielles (ou capteurs de vision dynamique), générant des événements asynchrones en réponse aux variations de luminosité, peuvent représenter d'autres solutions sensorielles prometteuses pour la détection à très faible consommation, notamment si elles sont couplées à des modèles neuronaux asynchrones [135].

Dans cette continuité, cette thèse introduit une architecture d'intelligence artificielle mixte appelée les *Primitive Neural Networks* (ou PrNN). L'objectif de cette architecture est de dépasser les limites du paradigme actuel en permettant l'exécution de tâches de reconnaissance¹ en temps réel sur systèmes embarqués, le tout avec un fonctionnement continu sur le très long terme². Pour ce faire, les PrNNs reposent sur un paradigme de co-évolution biologique, dans lequel le cerveau et ses organes sensoriels se sont développés de manière conjointe. La structure décrite dans la suite de cette contribution vise à permettre cette co-évolution entre une structure neuronale numérique traditionnelle et une interface matérielle analogique, tout en y ajoutant des mécanismes d'interconnexions neuronales similaires à ceux biologiques afin d'optimiser l'ensemble de la chaîne de traitement, de la perception jusqu'à l'inférence.

5.2 Une analogie hardware des processus biologiques

La structure générale du PrNN étant relativement complexe, nous faisons le choix de détailler étape par étape les réflexions et les mises en œuvre qui nous ont conduites à ce résultat.

Etape 1 : Effectuer un changement de paradigme

Afin de permettre l'évolution du système sensoriel de nos IAs, il faut comprendre que l'un des facteurs clé qui a permis cette possibilité aux systèmes biologiques réside dans leur paradigme du traitement analogique [136]. Contrairement aux systèmes numériques, qui reposent sur un échantillonnage discret et un traitement séquentiel sur l'ensemble de la donnée, les organismes biologiques exploitent des mécanismes analogiques continus et à très faible consommation, leur permettant de détecter et transformer les signaux en temps réel avec un minimum d'énergie.

En pratique, les données sensorielles sont d'abord captées et prétraitées par des organes spécialisés (tels que les yeux, les oreilles, la peau, le nez et la langue) avant d'être in-

1. Principalement acoustique dans cette étude, bien que pouvant être adaptée à d'autres tâches de reconnaissance sensorielle.

2. Supérieur à un an d'autonomie

interprétées par le cerveau. Pour chaque organe, cette étape de prétraitement consiste à extraire les primitives³ discriminantes du signal, permettant une forme d'extraction et de compression analogique de l'information utile⁴. Dans le cadre de la détection acoustique, exploiter ce paradigme nous amène à adopter des architectures inspirées de la cochlée.

Détaillée en section 3.4, la cochlée est un élément fondamental de l'oreille interne des mammifères et est responsable de la conversion des vibrations sonores en signaux neuronaux. Plutôt que de s'appuyer sur un échantillonnage numérique haute fréquence et une décomposition fréquentielle, nous allons donc imiter un comportement cochléaire en utilisant une série de filtres analogiques entre le signal d'entrée et le réseau de neurones [81, 137]. Cette approche permet de réduire significativement le volume d'informations numériques à échantillonner, tout en étant moins coûteuse qu'un prétraitement numérique⁵. Elle rend ainsi possible une écoute continue avec une consommation énergétique fortement réduite.

Etape 2 : Permettre l'optimisation énergétique de l'organe sensoriel

Bien que plus efficace que le traitement numérique, le traitement par filtrage analogique requiert néanmoins une certaine consommation d'énergie. Cette contrainte a entraîné une forme de sélection génétique des organismes, conduisant à l'évolution de leurs organes sensoriels, lesquels ont priorisés les stimuli en fonction de leur pertinence écologique et comportementale. Cette optimisation énergétique a restreint le domaine de sensibilité des organes, tout en introduisant la capacité des organismes à moduler cette sensibilité à l'intérieur de ce domaine, optimisant ainsi à la fois la performance perceptive et la consommation d'énergie.

En guise d'exemple, il conviendra de noter que certaines zones du corps contiennent une densité plus élevée de récepteurs sensoriels, et que certaines couleurs, odeurs ou saveurs sont plus saillantes selon leur importance pour la survie. Ce principe de sensibilité sélective est particulièrement évident pour la cochlée, dont la sensibilité fréquentielle est hautement spécifique à chaque espèce, reflétant un compromis évolutif entre une sensibilité large bande mais énergivore, et une audition restreinte mais économe en énergie [138, 139].

En nous appuyant sur ces observations, nous proposons d'optimiser la partie du PrNN correspondant au système cochléaire à l'aide d'un algorithme imitant la sélection naturelle. Ainsi, au lieu de fixer arbitrairement l'usage de nos filtres analogiques (une stratégie souvent surparamétrée), nous permettons aux PrNNs d'apprendre de manière autonome à allouer ces filtres en fonction de la tâche requise et de leurs contraintes énergétiques. Une telle approche permet de réduire significativement le nombre de filtres actifs (et donc la consommation énergétique) et de créer des IA aux capacités sensorielles variées,

3. Origine du terme : *Primitive Neural Networks*

4. Cette étape correspond en réalité à ce que les prétraitements numériques cherchent à reproduire et à transmettre au réseau de neurones. Elle permet d'introduire des opérations telles que des multiplications ou des divisions entre les différentes sources, ce qui n'est pas intrinsèquement réalisable dans le réseau de neurones numérique mais essentiel pour obtenir certaines caractéristiques physiques discriminantes, comme l'énergie d'un signal. Cette limite opératoire des réseaux de neurones numérique est la raison pour laquelle la grande majorité des modèles conventionnels d'IA acoustique performant une FFT en guise de prétraitement (cf. chapitre 1.2.1).

5. Une démonstration détaillée sera présentée en section 5.4

spécialisées pour leurs tâches, tout en s'appuyant sur une même structure matérielle.

Etape 3 : Extraction des primitives acoustiques

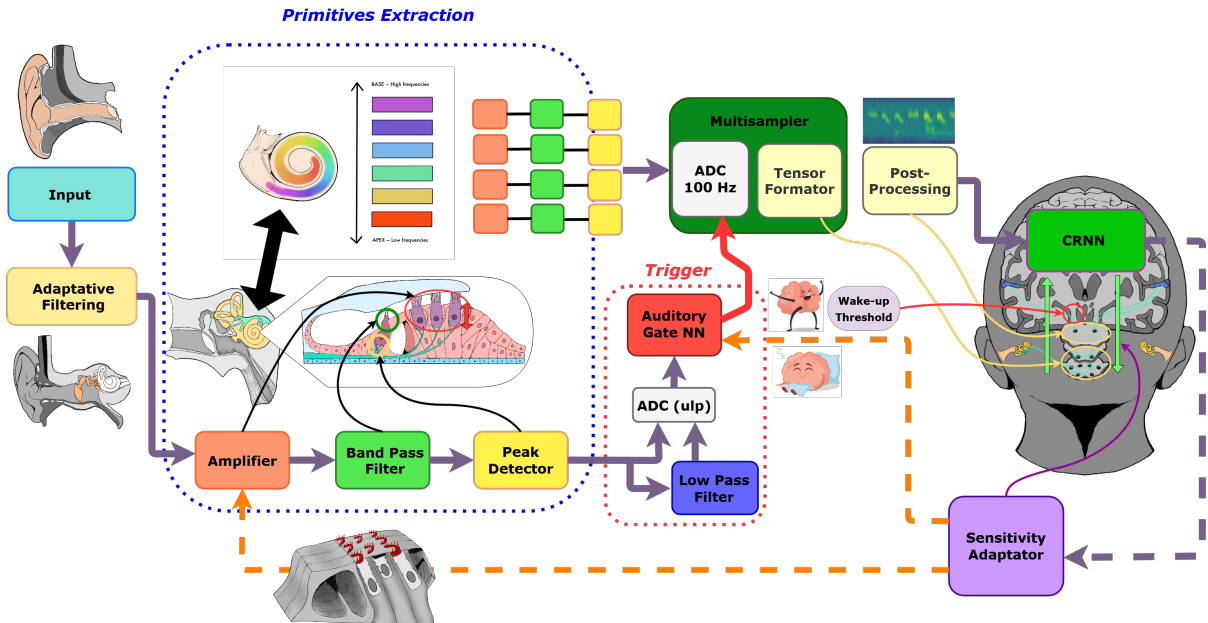


FIGURE 5.1 – Illustration de l'analogie entre le traitement auditif biologique et le design architectural des PrNNs dans le cadre de la reconnaissance acoustique.

La Figure 5.1 illustre les mécanismes biologiques impliqués dans la reconnaissance acoustique, de l'oreille au cerveau, ainsi qu'une proposition d'équivalence matérielle. L'architecture qui en résulte est celle défendue par cette thèse et correspond aux *Primitive Neural Networks*.

Dans ce système, le son est d'abord capté par un microphone (par exemple piézoélectrique) et filtré avec un gain adaptatif. Cette première étape est analogue au fonctionnement du tympan, des muscles et des osselets de l'oreille moyenne. Elle permet d'acquérir et d'amplifier dynamiquement le signal pour maintenir un niveau stable et non saturé.

Le signal est ensuite traité dans le domaine analogique par un banc de filtres imitant le comportement des cellules ciliées internes (IHC) et externes (OHC) de la cochlée⁶. Structuellement, chaque ressource de filtrage mise à disposition dans le PrNN est composée d'un amplificateur opérationnel (op-amp), suivi d'un filtre passe-bande et d'un détecteur de crête passe-bas.

L'op-amp utilisé en amont de ces ressources est une manière d'imiter la fonction des OHC. Biologiquement, les OHC sont reliées aux sorties neuronales du cerveau et permettent une amplification locale des différentes bandes de fréquence [112]. Cette modulation dynamique permet aux mammifères d'ajuster leur sensibilité fréquentielle au fil du temps, améliorant ainsi leur capacité à distinguer les sons dans un environnement donné. Nous reviendrons sur la manière dont sont contrôlés ces op-amps à la fin de cette section. Le

6. Cf. section 3.4

rôle des filtres passe-bande est quand à lui est similaire à celui des IHC. Chaque composante permet d'extraire l'une des primitives du signal acoustique. Ces informations sont ensuite transmises sous une forme compacte via un détecteur de crête, dans lequel seule l'enveloppe de chaque réponse fréquentielle est conservée⁷.

Dans les paragraphes suivants, nous ferons référence à cette partie architecturale comme le bloc de *Primitives Extraction* (PEX). Bien que l'organisation interne des ressources de filtrage soit structurellement fixée, l'utilisation ou non de ces ressources est optimisée par apprentissage.

Initialement, toutes les ressources sont utilisées, couvrant ainsi la totalité du spectre fréquentiel mis à disposition. Pendant l'entraînement, cet ensemble est progressivement élagué par un algorithme d'optimisation contraint par des limitations énergétiques et/ou de performances. Ces contraintes forcent la mise en place d'une co-optimisation entre la partie neuronales servant à la détection, nécessaire pour évaluer la performance globale, avec celle d'extraction de l'information, qui fixe l'énergie requise par le système. Ce processus permet à l'IA de ne conserver que les bandes de fréquences les plus informatives, quels que soient la tâche et le budget énergétique souhaités.

Etape 4 : Filtrage du flux informatif

Il serait erroné de croire qu'à ce stade, l'information extraite par la cochlée est directement transmise au cerveau. Si tel était le cas, le cerveau des mammifères serait soumis à un flux continu de stimuli à traiter, empêchant des actions réparatrices comme le sommeil d'opérer, et ce, même dans un environnement silencieux. En effet, bien que l'extraction sensorielle soit optimisée, l'énergie calculatoire nécessaire pour la détection reste conséquente et doit être utilisée avec parcimonie. Ce point clé a également été pris en compte dans la conception des PrNNs.

Avant d'atteindre les régions corticales, les signaux auditifs passent par une étape intermédiaire de traitement appelée *auditory gating* (AG), dans laquelle les stimuli redondants ou non pertinents sont filtrés le long de la voie auditive, notamment dans des structures sous-corticales comme le thalamus ou le cortex pariétal postérieur (PPC). Ce mécanisme de filtrage joue un rôle crucial dans la prévention de la surcharge sensorielle et garantit que seules les informations pertinentes sont transmises aux traitements neuronaux de plus haut niveau [141].

Bien qu'un filtrage auditif bio-inspiré puisse théoriquement être modélisé avec des *Spiking Neural Networks* (SNNs), la complexité pratique de tels systèmes a motivé une approche plus pragmatique sur le plan technologique. Ici, le filtrage auditif est réalisé par un petit réseau neuronal numérique fonctionnant à très basse fréquence (8Hz). Ce réseau compact, qui nécessite seulement quelques centaines d'opérations par inférence, traite un tenseur de taille fixe, construit à partir des primitives auditives extraites et de leurs versions filtrées en basse fréquence. Il détermine en permanence si le convertisseur analogique-numérique (ADC), utilisé pour échantillonner l'information à transmettre au réseau de

7. Bien que le signal biologique soit ensuite transmis sous forme de pulses discrets asynchrones [140], nous faisons le choix de rester ici dans le domaine continu par simplification architecturale.

neurones, doit être activé. Ainsi, seule cette partie de déclenchement (nommé *Trigger*) et le bloc de *Primitives Extraction* (PEX), ont un fonctionnement permanent. Le reste de l'architecture, plus calculatoire et consommatrice en énergie, n'est sollicitée que lorsque cela est nécessaire. L'apprentissage de ce réseau de neurones se fait après que les primitives auditives ont été sélectionnées.

Etape 5 : Échantillonnage numérique et reconnaissance sonore

Lorsque l'ADC est activé, la sortie du bloc PEX est échantillonnée à 100 Hz pendant une durée prédéfinie, variable selon l'application, puis combinée à d'autres données stockées dans des tampons locaux. Ces tampons correspondent aux mémoires internes des ADC utilisés pour la construction du tenseur d'entrée du bloc *Trigger*. Les ADC fonctionnent en permanence en mode ultra-basse consommation, mais disposent d'un tampon limité à 16 valeurs par canal, soit environ 0.16 s à 100 Hz. L'exploitation de ces tampons permet de reconstruire rétroactivement le signal précédant le déclenchement, garantissant ainsi qu'aucune information pertinente n'est perdue dans l'enregistrement transmis au réseau de neurones⁸.

La partie architecturale servant à la reconnaissance sonore est effectuée par un *Convolutional Recurrent Neural Networks* (CRNN), que l'on nommera *EcoPhonic*. Ce modèle combine des couches convolutives, pour l'extraction spatiale des caractéristiques, et des couches récurrentes, pour capturer les dépendances temporelles des motifs acoustiques. La structure neuronale d'*EcoPhonic* a été pensée de manière à pouvoir être embarquée sur des microcontrôleurs basse consommation comme le *STM32F413ZH* ou le *CC2652R*.

Aucun mécanisme de quantification ni d'accélérateur d'IA dédié n'a été utilisé dans cette étude. Ce choix découle principalement des contraintes de disponibilité matérielle et d'une volonté scientifique de concentrer l'attention du lecteur sur les contributions novatrices. Il convient de noter que l'impact de telles techniques est considéré comme modéré dans notre paradigme, étant donné que le réseau ne traite que des spectrogrammes contenant des informations pertinentes, et que ces événements sont supposés être rares.

Etape 6 : Adaptabilité aux changements d'environnement

Un défi majeur au déploiement de systèmes de reconnaissance autonomes pour des applications de terrain longue durée, est l'apparition sporadique de perturbations acoustiques inattendues près des capteurs. Ces perturbations peuvent être environnementales (conditions météorologiques extrêmes) ou biologiques (passage soudain d'un troupeau bruyant). Ces événements posent problème, non seulement en raison du risque accru de détections erronées, mais surtout parce qu'ils peuvent entraîner une activation fréquente et injustifiée de la chaîne de reconnaissance, réduisant considérablement l'autonomie énergétique attendue du système.

8. Bien que la fréquence d'échantillonnage de 100 Hz soit inférieure à la résolution temporelle du système auditif humain, estimée à environ 200 Hz [142], elle reste suffisante pour les applications pratiques. Cette valeur constitue un compromis adéquat pour le domaine acoustique humain, mais l'analyse de signaux à plus haute fréquence, comme ceux des chiroptères, nécessiterait une fréquence d'échantillonnage supérieure.

Compte tenu de leur rareté et de leur imprévisibilité, ces perturbations ne peuvent pas (et ne doivent pas) toutes être intégrées dans le jeu de données d’entraînement. Cette philosophie de conception s’oppose à une tendance courante dans la littérature qui consiste à élargir à la fois la taille des réseaux neuronaux et l’étendue de ce qui est appris, pour couvrir le plus de cas possibles.

Dans la nature, les organismes possèdent un cerveau de taille fixe, mais font preuve d’une adaptabilité remarquable, leur permettant d’obtenir des performances de détection quasiment indépendantes de leur environnement. Dans le domaine auditif, cela s’explique par la possibilité du cerveau à moduler dynamiquement la sensibilité cochléaire via les cellules ciliées externes (OHC). Ces modulations peuvent aller jusqu’à supprimer temporairement des bandes de fréquence normalement informatives mais devenues non pertinentes. En guise d’exemple, on peut citer notre capacité à ignorer un bruit de ronflement pour nous endormir, à ne pas prêter attention au bruit du vent ou de la pluie, ou encore à communiquer dans des environnements très bruyants.

Cette décision neuronale descendante (*top-down*⁹) est un mécanisme fondamental des systèmes biologiques, permettant un contrôle décisionnel arbitraire du système de reconnaissance sur le système de captation. Cet effet a longtemps été négligé dans les architectures d’IA, notamment pour des raisons de complexité d’intégration et d’utilité applicative lorsque l’environnement est relativement invarinant. On observe néanmoins un changement de tendance ces dernières années avec l’arrivée d’*Agents* IA, principalement basé sur les LLMs, concordant avec un cas applicatif devenu trop vaste pour des réseaux de neurones décisionnel fixe.

Nous notons que l’ajout d’un tel procédé décisionnel entraîne un surcoût calculatoire immédiat, particulièrement élevé dans l’exemple cité, ce qui pourrait sembler en contradiction avec l’objectif d’ultra-basse consommation. Cependant, la biologie nous montre que des avantages substantiels en termes d’efficacité énergétique comme de robustesse peuvent justifier certaines considérations plus adaptées, notamment lorsque l’IA fait face à un environnement imprévisible.

Pour reproduire ce principe biologique, l’architecture proposée intègre un composant adaptatif appelé *Adaptateur de Sensibilité*, qui agit directement sur la sortie du CRNN. Ce module analyse la cohérence temporelle des résultats de classification, renforce les détections récurrentes dans le temps, et atténue l’impact de changements soudains et incohérents (souvent liés à de faux positifs). Il évalue également la validité de chaque activation (c’est-à-dire, si le déclenchement numérique était justifié). Sur cette base, le système peut ajuster dynamiquement à la fois le seuil de déclenchement et l’amplification en entrée des filtres passe-bande. Ce mécanisme est contraint par un “effet ressort” bio-inspiré : en l’absence de perturbations persistantes, le système revient progressivement à sa configuration de base, sauf si de nouveaux stimuli contraires continuent de maintenir l’adaptation.

9. Du cerveau vers les organes sensoriels. Définie par opposition au transfert neuronal ascendant (*bottom-up*) des IA traditionnelles.

5.3 Une IA embarquable

Pour évaluer la faisabilité du déploiement de cette IA en embarqué, nous avons dû vérifier si les ressources de calcul fournies par des microcontrôleurs (MCU) basse consommation étaient suffisantes pour permettre des détections acoustiques.

Dans ce contexte, le **STM32F413ZH** conçu par *STMicroelectronics* a été retenu comme plateforme de test de notre approche. Ce microcontrôleur présente des contraintes matérielles strictes pour l’embarquement d’un réseau de neurones, avec **1.5 MB** de mémoire *Flash* et **320 KB** de *RAM*¹⁰. La mémoire *Flash* limite la taille maximale du modèle, notamment pour le stockage des poids, tandis que la *RAM* contraint la taille des activations et des buffers intermédiaires¹¹.

Les « ressources d’activation » d’un modèle correspondent aux besoins mémoires des caractéristiques intermédiaires, extraites par chaque couche neuronale. Lors de l’inférence, ces valeurs n’ont pas besoin d’être stockées au-delà de la couche suivante, permettant au MCU un recyclage des ressources mise en *RAM*. Néanmoins, ces limitations restreignent notamment le nombre de filtres convolutifs qui peut être utilisé et la taille du spectrogramme en entrée. De plus, plusieurs type courant de couches neuronales sont incompatibles avec l’environnement utilisé, principalement : les variantes bidirectionnelles des GRU, RNN et LSTM, les Transformeurs, les fonctions *Lambda*, et les couches personnalisées. Enfin, les couches récurrentes ne peuvent pas être quantifiées.

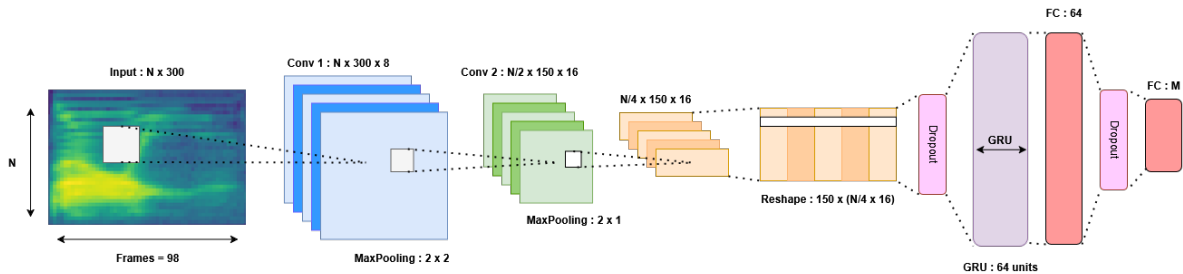


FIGURE 5.2 – Illustration de l’architecture du réseau de neurones *EcoPhonic*

À partir de ces contraintes, nous avons conçu l’architecture *EcoPhonic*, illustrée en Figure 5.2 et détaillée couche par couche dans le Tableau 5.1. Dans cette approche, nous utilisons des spectres audio correspondant à des fenêtres d’entrée de 3 secondes, plage utilisée ultérieurement pour notre tâche de reconnaissance acoustique d’oiseaux. N représente le nombre de primitives audio utilisées, qui varie entre 64 et 12 dans nos applications pratiques.

A titre de comparaison, le Tableau 5.2 propose une mise en perspective des ressources requises par notre modèle avec celles des différentes architectures présentées en état de

10. MB (*Mo* en unité française) : mégaoctet ; KB : kilooctet. Caractéristiques issues de la fiche technique du STM32F413.

11. L’utilisation de mémoires *RAM* ou *Flash* externes reste envisageable, mais au prix d’une consommation et d’une latence accrues, que nous cherchons à éviter.

l’art ¹². Les performances de ces modèles seront détaillées ultérieurement dans la partie III : *Résultats & Perspectives*. Dans ce tableau, l’utilisation de la RAM correspond à la mémoire d’activation maximale nécessaire, estimée par l’outil *STM32Cube.AI* (ces valeurs peuvent varier en fonction du matériel utilisé).

TABLE 5.1 – Résumé de l’architecture du modèle. La mémoire requise pour les paramètres de chaque couche est présentée avec 4 octets/valeur et sans quantification. Le coût calculatoire est présenté en *Multiply-Accumulate Operations* (MACs)

Layer	Out. Shape	Params.	Mem. (B)	MACs
Input	(-, 300, 64, 1)	0	0	0
Conv2D 3x3	(-, 300, 64, 8)	80	320	1 382 400
MaxPool2D	(-, 150, 32, 8)	0	0	0
Conv2D 3x3	(-, 150, 32, 16)	1 168	4 672	5 529 600
MaxPool2D	(-, 150, 16, 16)	0	0	0
Reshape	(-, 150, 256)	0	0	0
Dropout (0.25)	(-, 150, 256)	0	0	0
GRU (64)	(-, 64)	61 824	247 296	9 216 000
Dense (64)	(-, 64)	4 160	16 640	4 160
Dropout (0.3)	(-, 64)	0	0	0
Dense (10)	(-, 10)	650	2 600	650
Total		67 882	271 526	16 132 810

TABLE 5.2 – Exigences computationnelles et mémoire pour différents réseaux de neurones. La taille des spectrogrammes est propre à chaque modèle. Pour YOLOv8m, l’entrée devant être un multiple de 32, la taille est définie à 64×320 . Pour le modèle EcoPhonic, deux configurations sont présentées : une avec 64 filtres et une autre avec 12 filtres (12×300). Tous les résultats sont rapportés sans quantification (en supposant 4 octets par valeur). Nous rappelons que le GRU utilisé dans EcoPhonic ne peut pas être quantifié.

Modèle	Fenêtre (s)	MACs ($\times 10^6$)	Utilisation de la mémoire (MB)	
			Paramètres (Flash)	Activations (RAM)
SSL (AudioMAE)	10	2550.0	341.0	NA
Yolov8m	3	2000.0	98.7	2.58
Benbogat	5	1227.0	72.0	3.39
Perch	5	820.0	55.1	7.81
BirdNET v2.4	3	413.0	50.5	4.72
YamNet-1024	≈ 1	69.2	12.2	0.40
YamNet-256	≈ 1	25.2	0.53	0.40
EcoPhonic-64	3	16.1	0.27	0.16
EcoPhonic-12	3	4.6	0.10	0.03

En pratique, déterminer avec précision le nombre d’opérations *Multiply-Accumulate* (MAC) par inférence peut être difficile, en particulier lorsque l’architecture exacte du modèle n’est pas entièrement documentée ou accessible. Pour les modèles plus petits (comme les variantes *EcoPhonic* et *YAMNet*), les MACs ont été mesurés à l’aide du profileur intégré de

¹². Cf. Section 2.6

STM32Cube.AI. Pour les modèles restants, ce coup calculatoire a été estimé à l'aide des outils de profilage fournis par TensorFlow (*tf.profiler*) ou PyTorch (*torch.profiler*).

Au vu des limitations, seul le modèle proposé **EcoPhonic-64/12** est directement déployable. Le modèle **YamNET-256** reste compatible avec une quantification des poids (compression d'un facteur quatre).

5.4 Une structure permettant l'ultra basse consommation

Maintenant que nous avons détaillé l'architecture de notre IA et démontré qu'elle pouvait être mise en place dans un environnement embarqué, il nous reste à vérifier si l'objectif d'ultra-basse consommation peut être atteint.

Pour cette évaluation, nous ne prendrons pas en compte la consommation du microphone, car celle-ci constitue un coût fixe indépendant des choix architecturaux. En pratique, ce coût peut être négligé si l'on utilise un capteur piézoélectrique couplé à un amplificateur de tension, ou atteindre jusqu'à 250 μW avec un capteur MEMS analogique, ce qui, à lui seul, représente le budget énergétique maximal visé par nos travaux.

5.4.1 Énergie dépensée pour la partie analogique

Selon le paradigme défendu, la partie analogique est celle de notre architecture qui requiert le plus d'attention, car, contrairement à la partie numérique, elle est la seule devant maintenir un fonctionnement permanent.

Chaque filtre analogique utilisé dans cette partie peut être modélisé à l'aide d'un amplificateur opérationnel (AOP) stable en gain unitaire. Le coût énergétique des autres composants étant négligeable (car résistances élevées et condensateurs de qualité), la consommation totale de cette section dépend principalement du nombre et du type d'AOP sélectionnés. Afin d'orienter ce choix, le Tableau 5.3 présente un état de l'art des AOPs disponibles dans le commerce.

Au regard de notre utilisation, un produit gain-bande trop faible n'est pas envisageable, car il restreindrait le domaine fréquentiel exploitable, tandis qu'une valeur trop élevée serait incompatible avec l'objectif d'ultra-basse consommation. Pour ces raisons, les **MAX4464** ont été retenus pour leurs excellent compromis entre performance et efficacité énergétique, consommant seulement 1.08 μW par unité, pour un produit gain-bande de 40 kHz.

Pour chaque primitive extraite, le système requiert cinq AOPs : un pour le contrôle du gain, deux pour le filtrage passe-bande, un pour le détecteur de crête, et un pour le filtrage auxiliaire utilisé dans le mécanisme de déclenchement ; auquel s'ajoute l'utilisation de deux ADC à 1.8 μW .

Name	Gain-Bandwidth Product [kHz]	Current Supply [μ A]	Voltage Supply [V]	Slew Rate [V/ms]
TLV521	6	0.35	1.7 – 5.5	2.5
LPV801	8	0.3	1.6 – 5.5	2.1
TLV854x	8	0.5	1.7 – 3.6	4.5
TSU101	8	0.58	1.5 – 5.5	3
MAX40018	9	0.4	1.7 – 5.5	6.4
MCP6441	9	0.45	1.4 – 6.0	3
TSU111	11.5	0.9	1.5 – 5.5	2.5
MCP6041	14	0.6	1.4 – 6.0	3
MAX40007	20	0.7	1.7 – 5.5	12
MAX4464	40	0.6	1.8 – 5.5	20
OPA379	90	2.9	1.8 – 5.5	30
LTC2066	100	7.5	1.7 – 5.25	17.5
MAX40108	168	25.5	0.9 – 3.6	80
MAX40006	300	4.5	1.7 – 5.5	140
TLV9041	350	10	1.2 – 5.5	200
TSV6191	450	10	1.5 – 5.5	60
MP8102	600	7.5	1.8 – 5.5	100
LTC6258	1300	20	1.8 – 5.25	240

TABLE 5.3 – Comparatif des amplificateurs opérationnels stables en gain unitaire (*unity gain stable*).

Ainsi, le coût énergétique total par primitive peut être estimé à $9.0 \mu\text{W}$ ¹³. Deux amplificateurs et ADCs supplémentaires sont utilisés pour extraire un niveau de bruit du signal. En nous limitant à l'utilisation finale de **12 primitives**, nous pouvons estimer que la consommation totale de notre partie analogique s'élève à environ **114 μW** .

Cette consommation peut être réduite davantage en considérant d'autres méthodes d'implémentation.

L'alternative la plus courante consiste en une implémentation matérielle directe sous forme de circuit intégré (ASIC¹⁴ ou SoC¹⁵), plutôt qu'à l'aide de composants discrets. Ce type d'approche sort du périmètre de la thèse et fera l'objet du projet *ANR-DFG Sound IA*, dans le cadre de laquelle une thèse vient récemment de débiter. Certaines réalisations, fondées sur l'emploi d'oscillateurs en anneau pour le filtrage de bandes [81], ont déjà démontré un gain énergétique d'un ordre de grandeur.

Une autre voie de recherche, encore plus économe, repose sur l'utilisation de cochlées mécaniques auto-alimentées [143–145].

13. Il s'agit d'une estimation moyenne pour les bandes fréquentielles les plus hautes du domaine acoustique humain. En pratique, les AOP peuvent être dimensionnés en fonction de la bande fréquentielle visée (afin de consommer moins d'énergie aux basses fréquences). Cette estimation est volontairement généreuse afin d'englober les pertes dissipatives.

14. Application-Specific Integrated Circuit

15. System on Chip

5.4.2 Comparatif visuel avec l'approche numérique

Les spectrogrammes obtenus par méthode analogique présentent de légères différences par rapport à ceux générés par des méthodes numériques. Ces écarts, illustrés en Figure 5.3, s'expliquent par deux facteurs : l'ordre des filtres utilisés et le fenêtrage du signal.

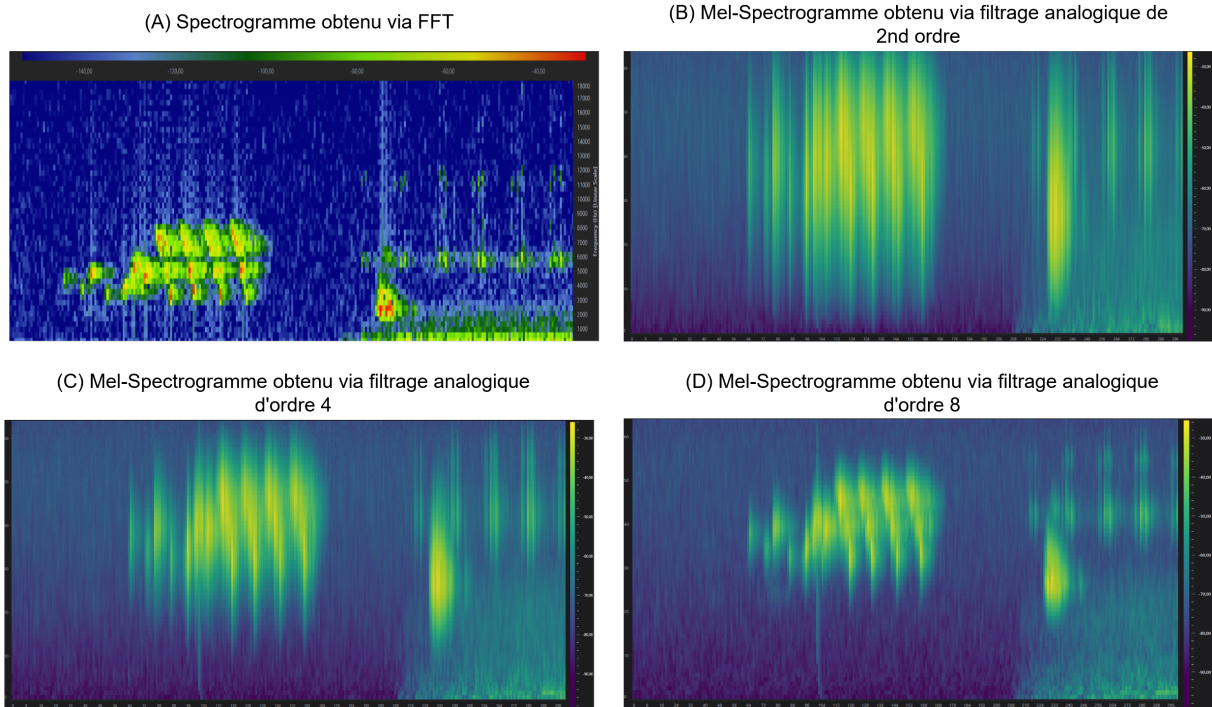


FIGURE 5.3 – Comparaison des spectres obtenus à l'aide de filtres d'ordre 2, 4 et 8, ainsi que via FFT. Ces spectres sont issus de vocalises d'oiseaux et comportent 64 bandes fréquentielles réparties le long d'une échelle MEL de 50 Hz à 18 kHz.

L'ordre utilisé pour les filtres passe-bande influence directement la qualité d'extraction d'une primitive. Plus l'ordre d'un filtre est élevé, moins les fréquences adjacentes influencent la bande courante, mais plus le nombre d'AOP requis augmente. Dans les calculs précédents, nous avons considéré un ordre 4, mais une analyse plus approfondie de l'impact de cet ordre sur les performances de détection sera présentée en *Section III : Résultats & Perspectives*.

La seconde distinction provient du mode de fenêtrage acoustique. Notre méthode analogique approxime l'énergie des bandes fréquentielles à l'aide d'un détecteur de pics passe-bas (LPPD). Ce filtrage peut engendrer un effet persistant de l'énergie lors de la transition du signal vers le silence. Dans sa version numérique, l'analyse repose sur une Transformée de Fourier Rapide (FFT) utilisant une fenêtre de taille fixe. Cette approche garantit qu'aucune énergie n'est présente pendant les phases de silence, mais peut également induire des phénomènes de Gibbs lors des transitions. Par ailleurs, les méthodes numériques peuvent bénéficier d'autres étapes de prétraitement telles que des pré-accentuation ou des normalisation par canal (PCEN).

5.4.3 Comparatif énergétique avec l'approche numérique

Bien que les caractéristiques issues de la méthode numérique soient, en théorie, plus précises que celles obtenues par extraction analogique de primitives, cette différence reste négligeable face au gain énergétique de la solution analogique. Pour le prouver, nous proposons de calculer le coût énergétique de l'approche numérique.

Acquisition du signal.

L'ADC doit numériser la sortie du microphone à haute fréquence (typiquement 36 kHz dans notre exemple). Selon la documentation du *STM32F413*, l'activation de l'ADC 12 bits consomme typiquement 1.6 mA pour activer la partie analogique. À 3.3 V, cela correspond à :

$$P_{\text{ADC}} = 3.3 \times 1.6 \text{ mA} \approx \mathbf{5.3 \text{ mW}}.$$

Transformée de Fourier.

Chaque trame de 10 ms (512 échantillons) est transformée en spectre de fréquence via une FFT de 512 points. La complexité d'une FFT radix-2¹⁶ est :

$$\frac{N}{2} \cdot \log_2(N) \text{ multiplications complexes, } N \cdot \log_2(N) \text{ additions complexes.}$$

Pour $N = 512$, cela représente 2304 multiplications complexes et 4608 additions complexes.

En opérations réelles équivalentes :

- 1 multiplication complexe = 4 multiplications réelles + 2 additions réelles.
- 1 addition complexe = 2 additions réelles.

On obtient ainsi environ 23 000 opérations arithmétiques par trame traitée.

Sur un ARM Cortex-M4, le *Floating Point Unit* (FPU), chargé du traitement des nombres en virgule flottante, exécute une opération en environ trois cycles d'horloge. En considérant une consommation typique de 112 μA /MHz à 3.3 V, on peut déduire que l'énergie par extraction est d'environ¹⁷ :

$$E_{\text{FFT}} = 23\,000 \times 3 \times \frac{112}{10^6} \times 3.3 \approx \mathbf{25.5 \mu\text{J}}.$$

En gardant le choix réaliste d'une trame toutes les 10 ms, cela correspond à 100 FFT/s, soit une puissance moyenne continue de :

$$P_{\text{FFT}} = E_{\text{FFT}} \times 100 \approx \mathbf{2.6 \text{ mW}}.$$

16. L'une des formes les plus courantes de l'algorithme de Cooley–Tukey.

17. Les observations pratiques sont plus proches de 10⁵ cycles/FFT ; on est donc optimiste.

Conversion en échelle Mel.

Le spectre de puissance sur 256 bins (jusqu'à 18 kHz) est projeté sur 64 bandes Mel via un banc de filtres triangulaires. Chaque bande est obtenue par une somme pondérée de plusieurs coefficients spectraux (poids pré-calculés).

Pour 64 filtres couvrant 256 bins :

$$64 \times 256 = 16\,384 \text{ multiplications et additions réelles par trame.}$$

Soit une énergie moyenne par conversion de :

$$E_{\text{Mel}} \approx 32\,700 \times 3 \times \frac{112}{10^6} \times 3.3 \approx \mathbf{36.3 \mu J}. \quad \Rightarrow \quad P_{\text{Mel}} \approx \mathbf{3.6 \text{ mW}}.$$

Total.

En additionnant les contributions :

$$P_{\text{total}} \approx 5.3 \text{ mW (ADC)} + 2.6 \text{ mW (FFT)} + 3.6 \text{ mW (Mel)} \approx \mathbf{11.5 \text{ mW}}.$$

Cette valeur est à mettre en perspective avec les 582 μW nécessaires à créer l'équivalent analogique. En considérant notre optimisation à 12 bandes (**114 μW**), l'implémentation analogique représente un **gain énergétique d'environ deux ordres de grandeur**.

5.4.4 Énergie dépensée pour la partie numérique

Dans notre configuration, la partie numérique a un besoin variable en ressources. C'est pourquoi, nous allons d'abord estimer la partie de ces besoins qui est **fixe** : il s'agit du coût d'inférence du petit réseau de neurones du *Trigger* (8 Hz) et du maintien du MCU en mode *Stop* le reste du temps.

Ce réseau de neurones requiert moins de 10^4 cycles par inférence, soit environ $8 \cdot 10^4$ cycles/s. Pour réaliser cette inférence en continu avec une dépense minimale, nous considérons un second microcontrôleur doté d'un mode ultra-basse consommation, à l'image de la série *STM32L4* de STMicroelectronics ou de l'*Apollo3 Blue* d'Ambiq. Ces MCU consomment typiquement de l'ordre de 6 à 84 $\mu\text{A}/\text{MHz}$ en mode *Run*, et environ 1 μA en mode veille profonde (Stop) sous une alimentation de 3.3 V. On peut alors estimer la puissance nécessaire au *Trigger* :

$$P_{\text{Trigger}} = \left(8 \cdot 10^4 \text{ Hz} \times \frac{6 \mu\text{A}}{10^6 \text{ Hz}} + 1 \mu\text{A} \right) \times 3.3 \text{ V} \approx 5 \mu\text{W}.$$

À ce coût fixe s'ajoute un coût **variable**, nécessaire lors d'une inférence du réseau principal. Ce coût correspond à l'échantillonnage de l'ADC principal à 100 Hz sur 12 canaux, ainsi qu'à l'inférence du réseau de neurones *EcoPhonic* pour la détection finale.

Pour l'acquisition, nous considérons un ADC externe ultra-basse consommation, tel que l'*AD4130-8*, permettant des acquisitions jusqu'à 16 canaux à 100 Hz pour un coût énergétique très faible¹⁸. Ce type de périphérique consomme typiquement 30 μA en fonctionnement continu (sous 3.3 V), et intègre une FIFO matérielle qui autorise le microcontrôleur hôte à rester en sommeil la majeure partie du temps d'échantillonnage.

$$E_{\text{ADC}} \approx 30 \mu\text{A} \times 3.3 \text{ V} \times 3 \text{ s} \approx 300 \mu\text{J}.$$

Le réseau de neurones principal, *EcoPhonic*, requiert environ 1.5×10^6 MAC/s, soit en moyenne 13.5×10^6 cycles pour une fenêtre de 3 secondes. Lorsqu'il est exécuté sur un *STM32F413*, la consommation active mesurée est de l'ordre de 3.4 mA à 20 MHz. Ainsi :

$$E_{\text{CRNN}} = \frac{13.5 \times 10^6}{20 \times 10^6} \times 3.4 \text{ mA} \times 3.3 \text{ V} \approx 7.5 \text{ mJ}.$$

En résumé, la partie numérique présente un coût **fixe** très faible, estimé à $P_{\text{fixe}} \approx 5 \mu\text{W}$, et un coût **variable** dépendant du taux d'occurrence de l'événement cible (environ 7.8 mJ par inférence de 3 s).

Pour un taux de présence de 5 % du temps (soit un déclenchement requis par minute), la puissance moyenne associée à cette composante variable devient :

$$P_{\text{variable}} \approx \frac{E_{\text{ADC}} + E_{\text{CRNN}}}{60} \approx \mathbf{130 \mu\text{W}}.$$

La consommation totale de la partie numérique devient alors :

$$P_{\text{total}} \approx P_{\text{fixe}} + P_{\text{variable}} \approx 5 \mu\text{W} + 130 \mu\text{W} \approx \mathbf{135 \mu\text{W}}.$$

5.4.5 Bilan

Notre système se compose de deux blocs complémentaires : une partie analogique à consommation constante, et une partie numérique dont l'activité dépend du contexte.

La partie analogique extrait en continu jusqu'à **12 primitives acoustiques** pour une consommation de **114 μW** . Cette opération équivaut fonctionnellement à l'extraction partielle d'un spectrogramme numérique, mais permet une réduction d'environ deux ordres de grandeur de l'énergie nécessaire.

La seconde composante, numérique, présente un coût énergétique variable selon la fréquence d'occurrence des événements détectés. En l'absence d'événements, cette consommation reste limitée à environ **5 μW** . Lorsqu'un événement est détecté, le traitement

¹⁸. L'usage de l'ADC interne du STM32F413ZH n'est pas adaptée à une utilisation ultra-basse consommation.

complet mobilise le microcontrôleur principal ainsi qu'un convertisseur externe, entraînant une consommation pouvant atteindre **2600 μW** en régime actif continu.

En pratique, dans un scénario où les événements d'intérêt sont présents **5% du temps**, la consommation moyenne totale du système est estimée à **249 μW** . Même en tenant compte du coût énergétique d'un microphone, ce niveau d'efficacité garantit une autonomie d'un an avec une pile AA Lithium. Sans implémentation ASIC, l'architecture se positionne ainsi à la frontière entre les systèmes à basse consommation et ceux à ultra-basse consommation.

Pour atteindre réellement le domaine de l'ultra-basse consommation ($< 100 \mu\text{W}$) tout en maintenant de bonnes performances, il faudra considérer l'une des alternatives citées pour la partie analogique continue (circuit intégré, oscillateurs en anneau ou cochlées mécaniques), suivie d'un microcontrôleur à l'architecture ultra-optimisée, doté d'un accélérateur neuronal dédié. Dans l'état de l'art actuel, peu de MCU parviennent à atteindre de telles efficacités énergétiques : on retrouve notamment les solutions de Syntiant (NDP102), GreenWaves (GAP8 et GAP9) et Analog Devices (MAX78000) [146]. Sous de telles implémentations, parvenir à réduire d'un ordre de grandeur supplémentaire la consommation énergétique des PrNNs semble crédible [81,147], une tendance partagée avec des variantes analogiques continues à base de SNNs pour les réseaux de neurones [148].

Contribution 3 : Construire, simuler et optimiser des PrNNs

6.1 Objectifs

Dans le cadre de cette thèse, nous avons développé un logiciel tout-en-un destiné à la conception d'architectures de type PrNN, à la simulation de leur fonctionnement, ainsi qu'à l'exécution des algorithmes d'optimisation en vue d'une implantation embarquée.

Ce simulateur poursuit un double objectif. À court terme, il sert d'outil d'ingénierie pour évaluer et tester de nouvelles structures matérielles avant leur fabrication. À plus long terme, il est conçu pour être accessible à un public non expert. L'objectif est de permettre à ses utilisateurs de choisir parmi des structures prédéfinies, puis de lancer des phases d'apprentissage afin de vérifier si les performances obtenues répondent à leurs attentes. L'outil intègre également un module spécifique pour l'étiquetage des données d'entraînement.

À l'issue de la phase d'apprentissage, le simulateur génère un noyau fonctionnel correspondant à l'architecture PrNN fournie en entrée. Ce noyau embarque la structure des réseaux de neurones, les poids appris, ainsi que les ressources matérielles à mobiliser pour l'extraction des primitives. Cette approche permet de produire des intelligences artificielles sur mesure, tout en garantissant une certaine modularité et une interchangeabilité des modèles au sein d'une même plateforme matérielle.

Cette contribution vise à détailler les fonctionnalités du simulateur développé, ainsi que le code associé aux différentes phases d'apprentissage.

6.2 Structure du code

L'architecture du code, se compose de trois modules principaux : l'interface graphique, le gestionnaire de structure (*PrNN Manager*) et le cœur algorithmique (*PrNN Core*). L'ensemble a été conçu de manière à ce qu'aucune donnée ne puisse être enregistrée en

dehors de l'ordinateur hôte.

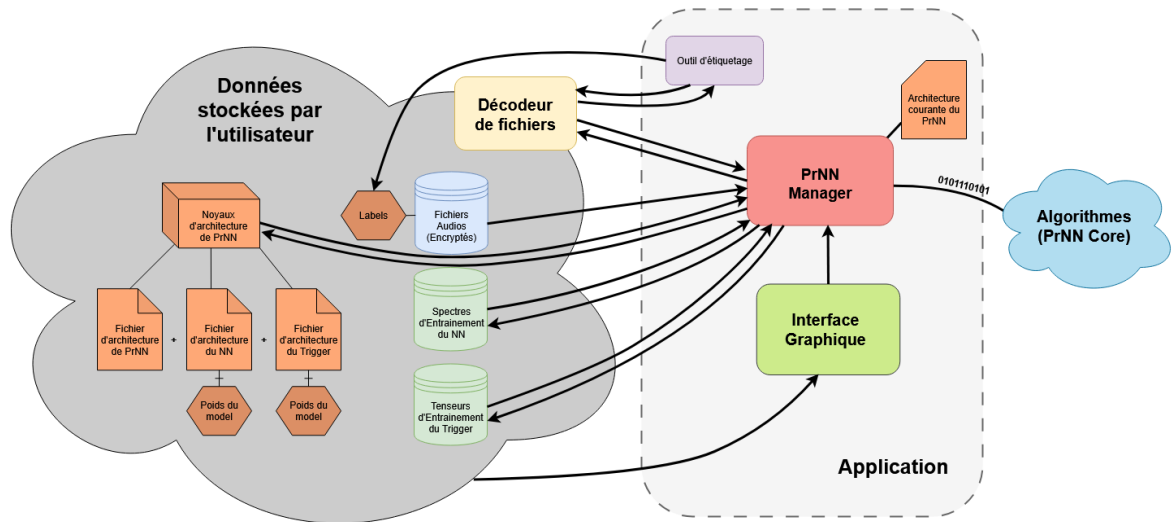


FIGURE 6.1 – Schéma bloc de la structure du code logiciel.

L'interface graphique constitue une surcouche purement visuelle permettant la communication avec le gestionnaire de structure. Bien qu'elle ne soit pas indispensable (le système pouvant fonctionner en mode API), elle s'avère très utile en pratique. Elle offre notamment la possibilité de visualiser et de modifier facilement l'état interne de l'architecture, de structurer sans code les réseaux de neurones à utiliser, d'observer les états intermédiaires entre les différentes composantes analogiques et numériques, ainsi que d'accéder à l'outil d'aide à l'étiquetage des fichiers.

Le gestionnaire de structure constitue le centre névralgique du code. Il est le seul module à posséder et à modifier les variables d'état de l'architecture. Le chargement d'une structure globale préenregistrée ou d'un réseau de neurones peut s'effectuer à partir des fichiers de noyau d'architecture PrNN sauvegardés au format *JSON*. Lors des phases de traitement, le gestionnaire communique directement avec un module de décodage (spécifique à l'utilisateur en cas de chiffrement) afin de transformer les données dans un format uniforme. Les tenseurs issus de la simulation des filtres analogiques, destinés à l'apprentissage des réseaux de neurones, sont enregistrés et stockés sur l'ordinateur hôte.

Les opérations de simulation et d'apprentissage, pouvant être lourdes et longues, peuvent être déléguées et parallélisées sur un centre de calcul externe. Le fait que le coeur algorithmique soit externalisé permet également de mettre à jour le code (améliorations ou corrections de bugs) sans nécessiter la réinstallation de l'application côté utilisateur.

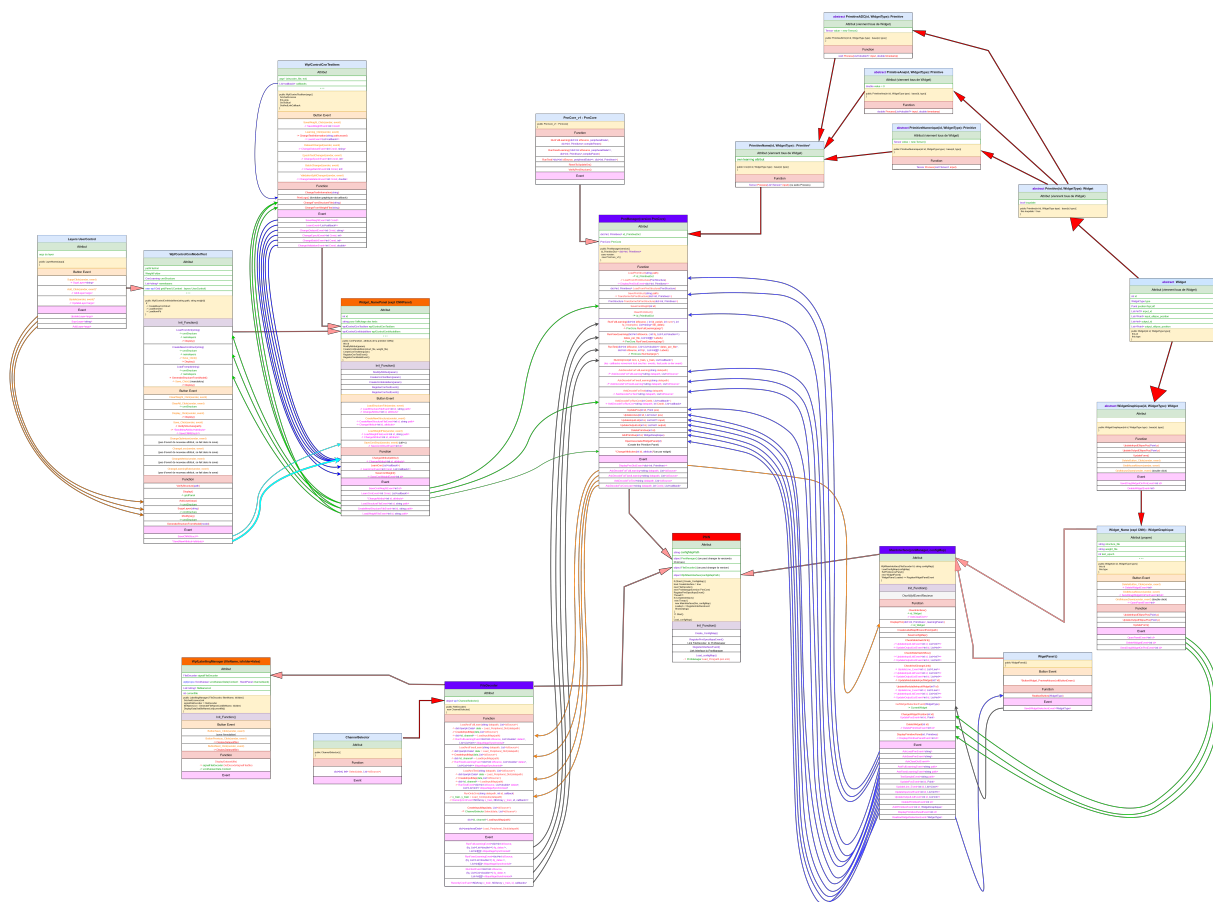


FIGURE 6.2 – Organisation exhaustive du code source. *Lisible en version pdf*

6.3 Interface graphique

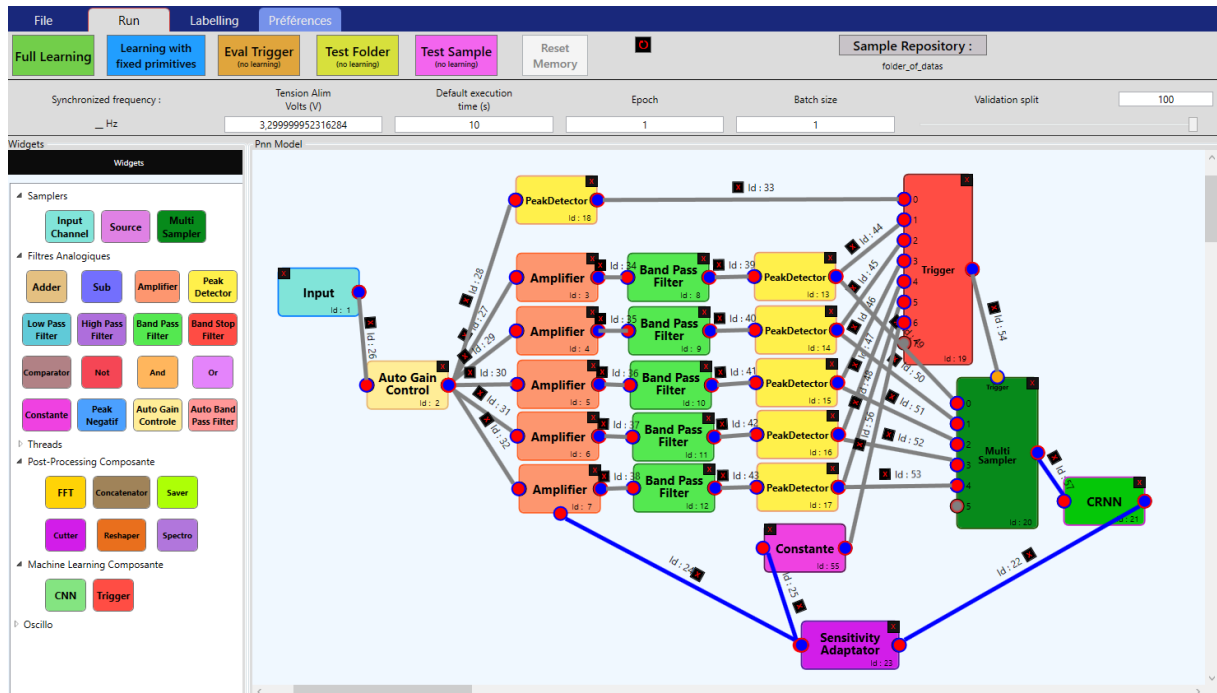


FIGURE 6.3 – Interface graphique du logiciel en phase de développement.

L'interface graphique, illustrée en Figure 6.3, se compose de trois zones principales. La zone centrale affiche la structure du PrNN dans son état courant. Chaque composant (ou *Widget*) peut y être déplacé et relié à un autre par des connexions analogiques ou numériques. Le type de connexion possible dépend du composant sélectionné.

Par exemple, le composant *Multi Sampler* (en vert foncé), qui agit comme un échantillonneur analogique, n'accepte en entrée (cercle rouge) que des connexions analogiques (lignes grises) et ne peut fournir en sortie (cercle bleu) qu'un bus numérique (lignes bleues). Les entrées marquées d'un cercle gris ne sont pas prises en compte lors de l'inférence. Elles apparaissent uniquement dans les composants capables de recevoir un nombre illimité d'entrées et servent de point d'ajout pour de nouvelles connexions. Un double-clic sur un composant ouvre un panneau affichant ses paramètres internes, qui peuvent alors être modifiés.

Le panneau situé à gauche regroupe les composants disponibles. L'ajout d'un composant s'effectue par un simple glisser-déposer depuis ce panneau vers la zone centrale. Cliquer sur la croix rouge d'un composant permet de le supprimer. Chaque élément possède un identifiant unique garantissant la distinction de ses paramètres par rapport aux autres.

En haut de l'interface se trouve le panneau des commandes, organisé en onglets regroupant diverses fonctions : sauvegarde ou chargement d'une structure, lancement d'apprentissages ou de tests, accès à l'outil d'étiquetage, et modification de paramètres par défaut.

Les entrées du réseau peuvent être de deux types :

- **Les Inputs** : fournissent en sortie le contenu d'un canal issu d'un fichier.
- **Les Sources** : génèrent en sortie des signaux prédéfinis.

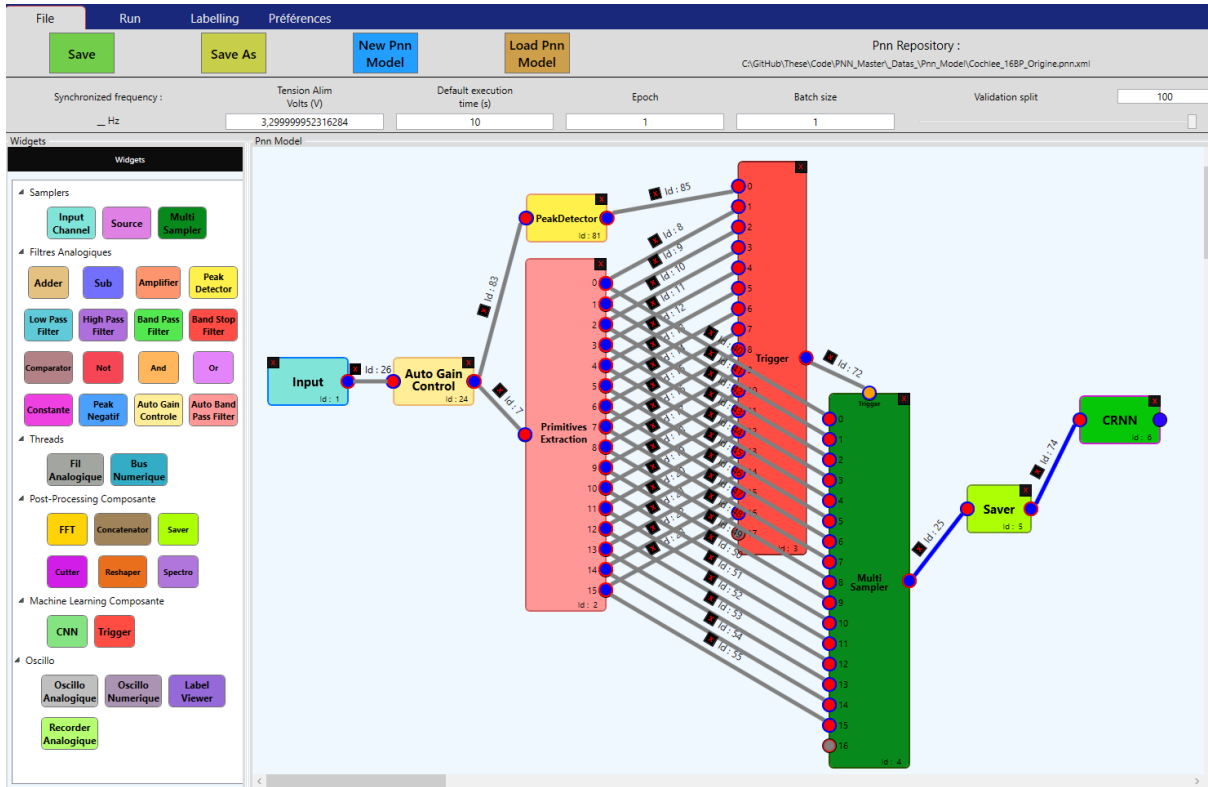


FIGURE 6.4 – Interface graphique du logiciel en phase d'utilisation réelle.

Pour des raisons ergonomiques, la représentation réelle d'une structure PrNN diffère légèrement de la phase de développement (Figure 6.4). En pratique, certains composants destinés à l'extraction de primitives (amplificateur opérationnel, filtre passe-bande, détecteur de crête) peuvent être regroupés en un seul élément. De plus, le composant *Sensitivity Adaptator* n'apparaît pas dans l'interface : lorsqu'il est requis, il est ajouté automatiquement à la structure mais reste invisible pour l'utilisateur. Ces choix visent à alléger l'affichage et à réduire le risque d'erreurs humaines.

Pour fonctionner correctement, la structure de base d'une architecture PrNN doit correspondre au schéma de la Figure 6.5. Un apprentissage nécessite une structure composée au minimum de :

- une entrée *Input*,
- un bloc *Primitives Extraction*,
- un *Trigger*,
- un *Multisampleur* (ADC),
- un *CRNN*.

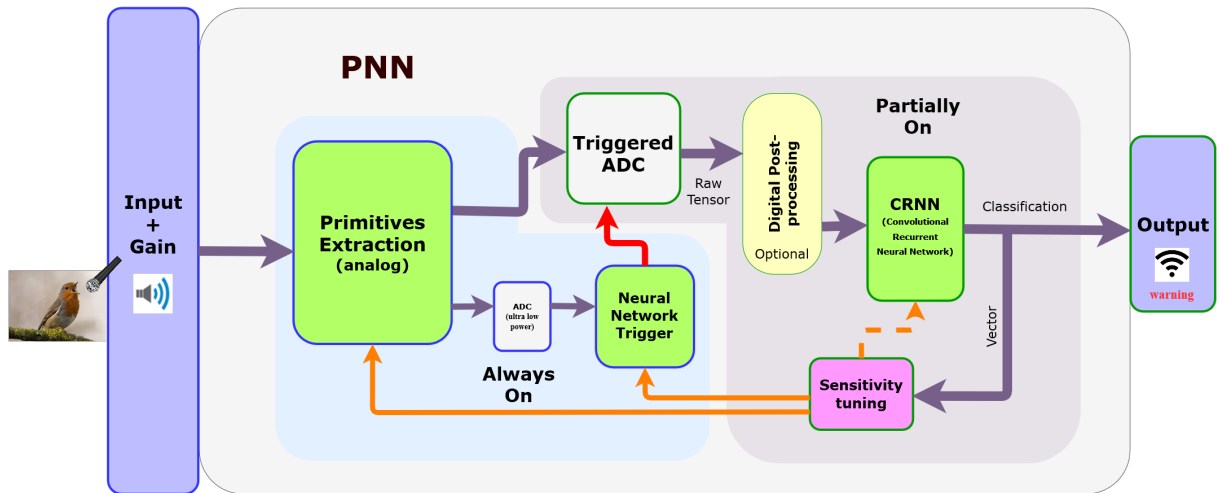


FIGURE 6.5 – Schéma de l'architecture PrNN.

Il est possible de disposer de plusieurs canaux d'entrée et de plusieurs réseaux neuronaux. Ce choix ouvre la voie à une future intégration de la localisation spatiale, en complément de la reconnaissance sonore.

Le bloc *CRNN* peut être structuré de manière quasi libre : il n'est pas obligatoire de comporter des couches récurrentes ou convolutionnelles, bien que cela soit recommandé. Un double-clic sur cet élément ouvre le panneau de configuration du réseau (Figure 6.6), permettant de concevoir un réseau de neurones sans écrire de code ou de charger un modèle existant. Les fichiers de structure sont enregistrés au format XML.

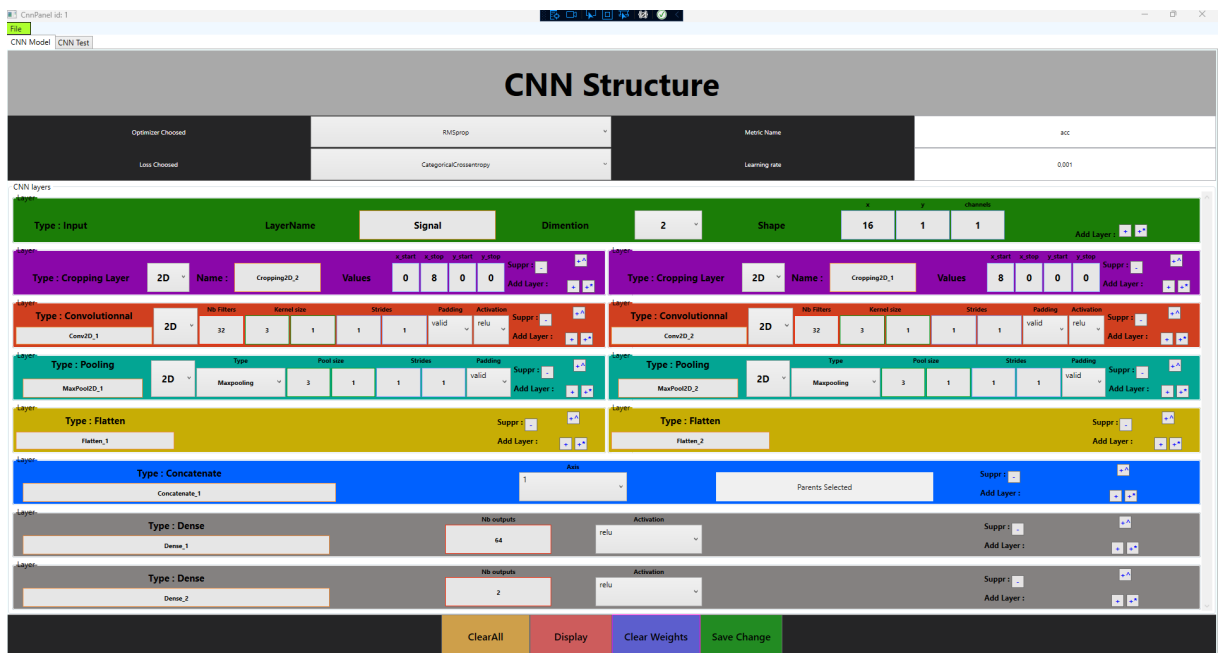


FIGURE 6.6 – Interface dédiée au réseau de neurones.

En changeant d'onglet, l'utilisateur accède au panneau d'entraînement (Figure 6.7), dédié à l'apprentissage isolé du réseau. Ce type de test est généralement couplé à l'utilisation d'éléments comme le *Saver*, qui permet d'enregistrer les tenseurs en entrée et ainsi de constituer un jeu de données issu de l'extraction analogique.

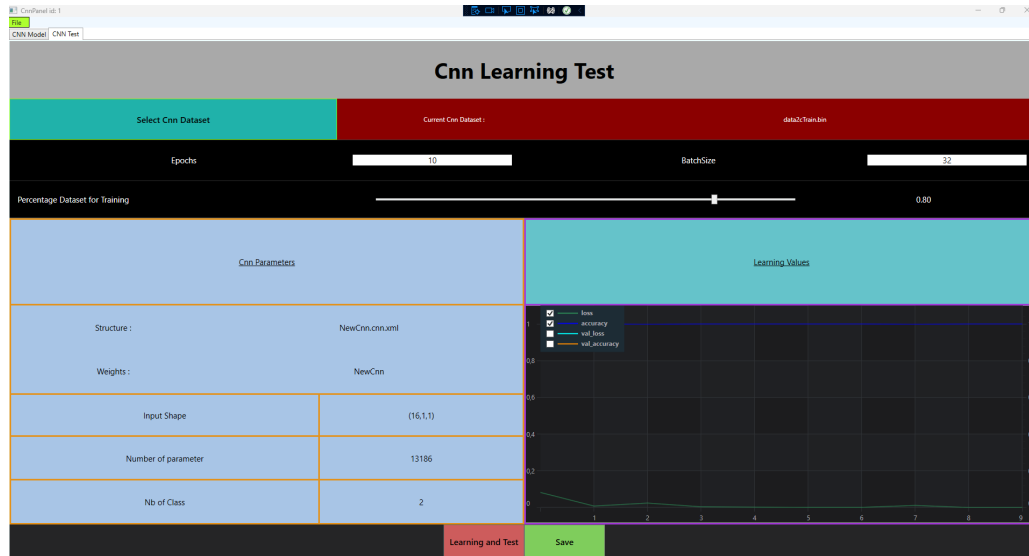


FIGURE 6.7 – Interface pour l'apprentissage isolé d'un réseau de neurones.

Pour visualiser le comportement de la structure, il est également possible de connecter les composants à des oscilloscopes. Ceux-ci peuvent être numériques, comme illustré en Figure 6.8, ou analogiques. Ces oscilloscopes permettent aussi de visualiser à quel classe acoustique correspond le signal relevé.

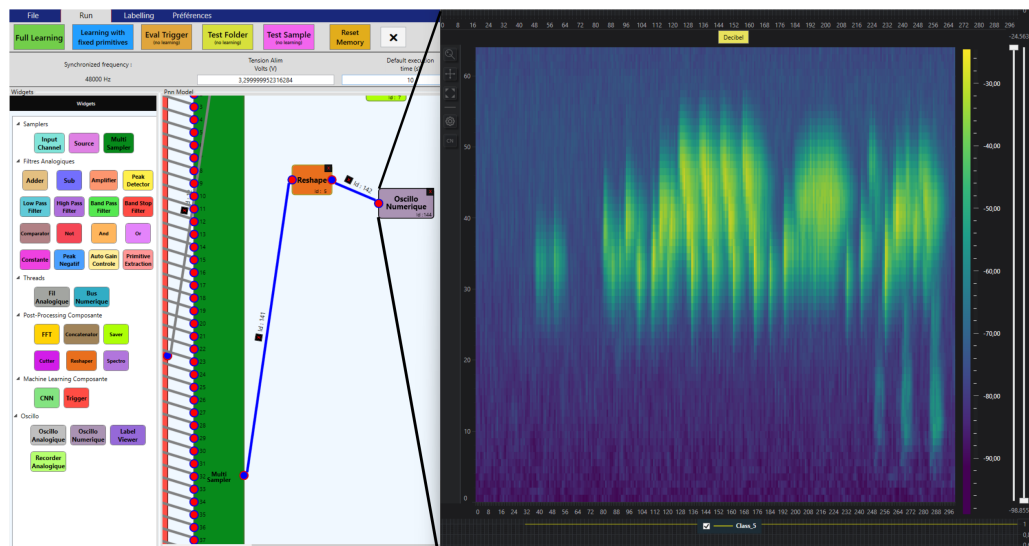


FIGURE 6.8 – Panneau de visualisation d'un oscilloscope dans le logiciel.

Enfin, l'outil d'aide à l'étiquetage est présenté en Figure 6.9. Il offre plusieurs modes de

visualisation, ainsi que la possibilité d'écouter certains segments audio et de supprimer le fichier. Lorsqu'un dossier est traité, les labels sont enregistrés automatiquement lors du passage d'un fichier à l'autre.

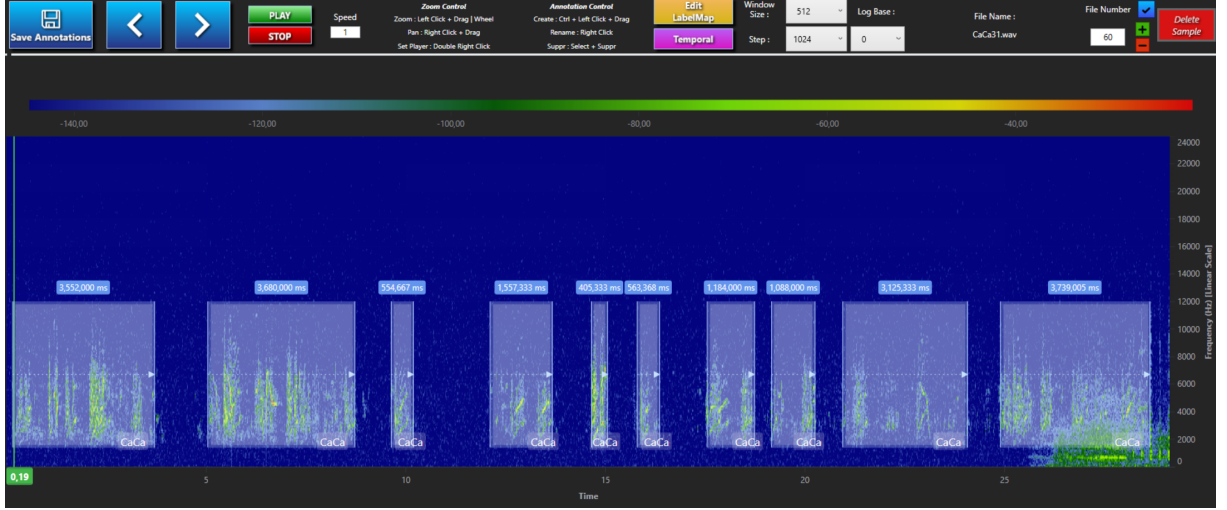


FIGURE 6.9 – Représentation de l'outil d'étiquetage des signaux audio.

6.4 Simulation des filtres

Cette section décrit des équations utilisées afin de modéliser le comportement analogique des filtres à partir d'enregistrements numériques. Les systèmes décrits étant déjà connus dans la littérature, il ne s'agit pas d'une contribution en soi, simplement d'un rappel mathématique justifiant le code utilisé en simulation.

6.4.1 Étude des filtres BP, LP et HP

Étude dans le domaine analogique

Dans notre implémentation, nous obtenons nos filtres passe-bande (BP) par une mise en série de filtres passe-bas (LP) et passe-haut (HP) de type Butterworth.

$$H^{BP}(s) = H^{LP}(s)H^{HP}(s)$$

Pour cette raison, nous étudierons d'abord le prototype passe-bas de Butterworth, puis nous en déduirons le passe-haut par transformation fréquentielle, et donc le passe-bande utilisé.

Par définition, le gain (normalisé) d'un filtre Butterworth passe-bas d'ordre n est :

$$|H(j\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}},$$

où $\omega = 2\pi f$ (en $\text{rad} \cdot \text{s}^{-1}$) est la pulsation et ω_c la pulsation de coupure.

Cette réponse est monotone (pente asymptotique de $-20n$ dB/décade à partir de ω_c) et maximale plate en $\omega = 0$, sans zéros finis. On adopte donc un modèle *tout-pôles* (numérateur constant) de la forme $H(s) = K/B_n(s)$, normalisé en $H(0) = 1$, d'où :

$$|H(j\omega)|^2 = \frac{K^2}{|B_n(j\omega)|^2} = \frac{K^2}{B_n(j\omega) B_n(-j\omega)}.$$

Cette formulation revient à choisir B_n tel que, à une constante de normalisation près,

$$B_n(j\omega) B_n(-j\omega) = 1 + \left(\frac{\omega}{\omega_c}\right)^{2n}.$$

En posant $p \stackrel{\text{déf}}{=} \frac{j\omega}{\omega_c}$, on obtient :

$$1 + \left(\frac{\omega}{\omega_c}\right)^{2n} = 1 + (-1)^n p^{2n}.$$

Les racines de B_n sont donc les points satisfaisant :

$$1 + (-1)^n p^{2n} = 0 \iff p^{2n} = (-1)^{n+1}.$$

Deux cas se présentent :

- n **pair** : $(-1)^{n+1} = -1 \Rightarrow p^{2n} = -1 \Rightarrow p_m = e^{j\frac{(2m+1)\pi}{2n}}$, $m = 0, \dots, 2n-1$.
- n **impair** : $(-1)^{n+1} = +1 \Rightarrow p^{2n} = 1 \Rightarrow p_m = e^{j\frac{m\pi}{n}}$, $m = 0, \dots, 2n-1$.

Dans *les deux cas*, on obtient $2n$ racines uniformément espacées sur le cercle unité : la parité de n ne change qu'une *rotation globale* du motif (décalage angulaire de $\pi/(2n)$), pas sa structure équidistante.

En revenant à la variable de Laplace $s = \omega_c p$, ces racines sont :

$$s_m = \omega_c p_m, \quad |s_m| = \omega_c,$$

c'est-à-dire $2n$ points équidistants sur le cercle de rayon ω_c .

Pour un système linéaire, invariant dans le temps (LTI) et continu, la réponse impulsionnelle s'écrit :

$$h(t) = \mathcal{L}^{-1}\{H(s)\}.$$

Pour assurer *causalité* ($h(t) = 0$ pour $t < 0$) et *stabilité BIBO* (entrée bornée $\Rightarrow$ sortie bornée), on sait démontrer qu'il ne faut retenir que les pôles dans le *demi-plan gauche* ($\Re\{s\} < 0$), soit exactement les n points d'argument dans $[\pi/2, 3\pi/2]$. Une indexation standard (et indépendante de la parité de n) est de poser :

$$\phi_k = \frac{\pi}{2} + \frac{(2k-1)\pi}{2n}, \quad k = 1, \dots, n,$$

et

$$s_k = \omega_c e^{j\phi_k}, \quad k = 1, \dots, n.$$

Comme $(s_k)_{k=1}^n$ sont les n racines du polynôme B_n , on a :

$$B_n(s) = \prod_{k=1}^n (s - s_k).$$

Et puisque l'on souhaite la normalisation, on a $H(0) = 1$, ce qui fixe la constante K .

$$K = B_n(0) = \prod_{k=1}^n (-s_k) = \omega_c^n,$$

Ainsi, la fonction de transfert s'écrit :

$$H(s) = \frac{\omega_c^n}{\prod_{k=1}^n (s - s_k)} = \prod_{k=1}^n \frac{\omega_c}{s - \omega_c e^{j\phi_k}} = \prod_{k=1}^n \frac{\omega_c}{s - \omega_c e^{\frac{j(2k+n-1)\pi}{2n}}}.$$

De plus, en notant s_k^* le conjugué de s_k , on remarque que :

$$(s - s_k)(s - s_k^*) = s^2 - 2\Re\{s_k\}s + |s_k|^2 = s^2 - 2\omega_c \cos \phi_k s + \omega_c^2.$$

Donc, en regroupant tous les pôles conjugués (s_k, s_k^*) et en posant

$$\zeta_k = -\cos \phi_k = \sin\left(\frac{(2k-1)\pi}{2n}\right),$$

on obtient des sections du second ordre à coefficients réels.

Si n est pair,

$$H(s) = \prod_{k=1}^{n/2} \frac{\omega_c^2}{s^2 + 2\zeta_k \omega_c s + \omega_c^2},$$

et si n est impair,

$$H(s) = \frac{\omega_c}{s + \omega_c} \prod_{k=1}^{\frac{n-1}{2}} \frac{\omega_c^2}{s^2 + 2\zeta_k \omega_c s + \omega_c^2}.$$

Le filtre passe-haut est dérivé du passe-bas par la transformation fréquentielle $s \mapsto \frac{\omega_c^2}{s}$, qui inverse la réponse en fréquence.

Passage au numérique

On approxime l'intégration continue du filtre par la règle trapézoïdale :

$$y[m] - y[m-1] \approx \frac{T}{2}(x[m] + x[m-1]),$$

où $T = 1/F_s$ est la période d'échantillonnage, $x[m]$ est le m -ième échantillon et $y[m]$ est la réponse du filtre à cet échantillon.

En appliquant la transformée en Z (états initiaux nuls) à cette approximation, on obtient :

$$Y(z)(1 - z^{-1}) = \frac{T}{2}(1 + z^{-1})X(z) \quad \Rightarrow \quad \frac{Y}{X} = \frac{T}{2} \frac{1 + z^{-1}}{1 - z^{-1}}$$

Or, en continu, la transformée de Laplace dit que $\mathcal{L}\{f x\} = X(s)/s$, donc l'intégrateur vérifie :

$$H(s) = \frac{Y(s)}{X(s)} = \frac{X(s)}{sX(s)} = \frac{1}{s}$$

Ainsi, l'intégrateur discret correspondant à l'intégrateur continu (transformation bilinéaire ou méthode de Tustin), s'écrit :

$$\frac{1}{s} \rightarrow \frac{T}{2} \frac{1 + z^{-1}}{1 - z^{-1}} \quad \Leftrightarrow \quad \boxed{s \rightarrow 2F_s \frac{1 - z^{-1}}{1 + z^{-1}}}$$

Dès lors, la fonction de transfert discrète associée à un filtre analogique est

$$H_{\text{discret}}(z) = H_{\text{analogique}} \left(2F_s \frac{1 - z^{-1}}{1 + z^{-1}} \right).$$

L'une des propriétés majeure de cette transformation est que l'image de l'axe imaginaire $s = j\omega$ est le cercle unité $|z| = 1$ et que celle du demi-plan complexe gauche ($\Re(s) < 0$) est à l'intérieur du cercle unité ($|z| < 1$). Cette propriété transporte la stabilité analogique vers une stabilité numérique.

Cependant, puisque chaque point $s = j\omega$ a pour image un point $z = e^{j\Omega}$, cela entraîne aussi une correspondance fréquentielle non linéaire (ou *warping*).

$$s = 2F_s \frac{1 - z^{-1}}{1 + z^{-1}} \quad \Rightarrow \quad s(1 + z^{-1}) = 2F_s(1 - z^{-1}) \quad \Rightarrow \quad s + sz^{-1} = 2F_s - 2F_s z^{-1}$$

$$\Rightarrow (s + 2F_s)z^{-1} = 2F_s - s \quad \Rightarrow \quad z^{-1} = \frac{2F_s - s}{2F_s + s} \quad \Rightarrow \quad z = \frac{2F_s + s}{2F_s - s}$$

D'où :

$$\boxed{z = \frac{1 + j \frac{\omega}{2F_s}}{1 - j \frac{\omega}{2F_s}} = e^{j\Omega}}$$

En utilisant les propriétés $\arg(1 + jx) = \arctan(x)$ et $\arg(1 - jx) = -\arctan(x)$, on trouve que la correspondance entre la pulsation analogique ω et la pulsation numérique Ω (rad/échantillon) est :

$$\Omega = 2 \arctan\left(\frac{\omega}{2F_s}\right) \iff \omega = 2F_s \tan\left(\frac{\Omega}{2}\right).$$

Ainsi, pour compenser cet effet et maintenir la fréquence de coupure numérique f_c sélectionnée (soit $\Omega_c = 2\pi f_c/F_s$), on *pré-gauchit* la coupure analogique (on remplace ω_c par ω'_c) :

$$\boxed{\omega'_c = 2F_s \tan\left(\frac{\pi f_c}{F_s}\right)}$$

6.4.2 Implémentation Code

Pour des raisons de stabilité numérique et de robustesse aux erreurs d'arrondi, les filtres sont implémentés sous forme de cascade de sections du second ordre (SOS).

D'après notre étude, si n est pair¹, on a :

$$H(s) = \prod_{k=1}^{n/2} H_k(s) \quad , \quad H_k^{\text{LP}}(s) = \frac{(\omega_c^{\text{LP}})^2}{s^2 + 2\zeta_k \omega_c^{\text{LP}} s + (\omega_c^{\text{LP}})^2}, \quad H_k^{\text{HP}}(s) = \frac{s^2}{s^2 + 2\zeta_k \omega_c^{\text{HP}} s + (\omega_c^{\text{HP}})^2},$$

ce qui nous donne en version numérique :

$$H_k^{\text{LP}}(z) = \frac{\omega_c^{\text{LP}^2}}{\left(2F_s \frac{1-z^{-1}}{1+z^{-1}}\right)^2 + 2\zeta_k \omega_c^{\text{LP}'} \left(2F_s \frac{1-z^{-1}}{1+z^{-1}}\right) + \omega_c^{\text{LP}^2}}.$$

$$\Leftrightarrow H_k^{\text{LP}}(z) = \frac{(\omega_c^{\text{LP}'}(1+z^{-1}))^2}{(2F_s(1-z^{-1}))^2 + 4\zeta_k \omega_c^{\text{LP}'} F_s(1-z^{-1})(1+z^{-1}) + (\omega_c^{\text{LP}'}(1+z^{-1}))^2}$$

Ainsi, pour modéliser un filtre d'ordre n , on peut créer $n/2$ sections telles que :

$$H(z) = \frac{Y(z)}{X(z)} = \prod_{k=1}^{n/2} H_k(z) \quad \text{avec} \quad H_k(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{b_0^{(k)} + b_1 z^{-1} + b_2^{(k)} z^{-2}}, \quad \text{pour } k = 1, \dots, \frac{n}{2}$$

Si n est impair, on a la variante :

$$H(z) = H_0(z) \prod_{k=1}^{\frac{n-1}{2}} H_k(z) = \frac{a_0^{(0)} + a_1^{(0)} z^{-1}}{1 + b_1^{(0)} z^{-1}} \prod_{k=1}^{\frac{n-1}{2}} H_k(z)$$

1. le cas n impair se traite comme pour le cas pair à l'ordre $n - 1$ mais en ajoutant la multiplication d'une section du premier ordre

Pour un LP, les coefficients $(a_0, a_1, a_2, a_0^{(0)}, a_1^{(0)})$ sont :

$$a_0 = a_2 = \left(\omega_c^{LP'}\right)^2, \quad a_1 = 2 \left(\omega_c^{LP'}\right)^2, \quad a_0^{(0)} = a_1^{(0)} = \frac{\omega_c^{LP'}}{2F_s + \omega_c^{LP'}}.$$

Tandis que pour un HP :

$$a_0 = a_2 = 4F_s^2, \quad a_1 = -8F_s^2, \quad a_0^{(0)} = \frac{2F_s}{2F_s + \omega_c^{HP'}}, \quad a_1^{(0)} = -\frac{2F_s}{2F_s + \omega_c^{HP'}}.$$

Dans les deux cas (LP ou HP), les coefficients $(b_0^{(k)}, b_1, b_2^{(k)}, b_1^{(0)})$ sont $(\omega'_c = \omega_c^{LP'} \text{ ou } \omega_c^{HP'})$:

$$b_0^{(k)} = 4F_s^2 + 4F_s \zeta_k \omega'_c + \omega_c'^2, \quad b_1 = 2\omega_c'^2 - 8F_s^2, \quad b_2^{(k)} = 4F_s^2 - 4F_s \zeta_k \omega'_c + \omega_c'^2, \quad b_1^{(0)} = \frac{\omega'_c - 2F_s}{\omega'_c + 2F_s}.$$

En posant $D_k(z^{-1}) = b_0^{(k)} + b_1 z^{-1} + b_2^{(k)} z^{-2}$ et $N_k(z^{-1}) = a_0 + a_1 z^{-1} + a_2 z^{-2}$, on peut réécrire les fonctions de transfert :

$$H_k = \frac{Y_k}{X} = \frac{N_k}{D_k} \Rightarrow D_k y_k = N_k x.$$

Pour simuler un filtre, nous allons donc stocker $\left\lfloor \frac{n}{2} \right\rfloor$ sections de filtrage ayant chacune pour fonction (conditions initiales nulles) :

$$b_0^{(k)} y_k[m] + b_1 y_k[m-1] + b_2^{(k)} y_k[m-2] = a_0 x[m] + a_1 x[m-1] + a_2 x[m-2]$$

$$\Rightarrow \boxed{y_k[m] = \frac{1}{b_0^{(k)}} \left(a_0 x[m] + a_1 x[m-1] + a_2 x[m-2] - b_1 y_k[m-1] - b_2^{(k)} y_k[m-2] \right)}.$$

Note : si n est impair, on ajoute une section ayant pour fonction :

$$y[m] = a_0^{(0)} x[m] + a_1^{(0)} x[m-1] - b_1^{(0)} y[m-1].$$

Puis, on les met en cascade comme dans l'algorithme 1 pour obtenir la fonction de filtrage générale.

Algorithm 1 Calcul de $y[m]$ (cascade de sections)

Require: {Section[0], ..., Section[$\lceil n/2 \rceil - 1$]}

```

1: function FILTRE( $x[m]$ )
2:    $y \leftarrow x[m]$ 
3:   for  $k \leftarrow 0$  to  $\lceil n/2 \rceil - 1$  do
4:      $y \leftarrow \text{Section}[k].\text{filtre}(y)$ 
5:   end for
6:   return  $y$ 
7: end function
```

On obtient bien :

$$\begin{aligned}
D_1 y_1 &= N_1 x, \quad D_2 y_2 = N_2 y_1, \quad \dots, \quad D_{\frac{n}{2}} y_{\frac{n}{2}} = N_{\frac{n}{2}} y_{\frac{n}{2}-1} \\
\Rightarrow y_1 &= \frac{N_1}{D_1} x, \quad D_2 D_1 y_2 = N_2 N_1 x, \quad \dots, \\
\Rightarrow D_{\frac{n}{2}} \dots D_2 D_1 y_{\frac{n}{2}} &= N_{\frac{n}{2}} \dots N_2 N_1 x \\
\Rightarrow Y_{\frac{n}{2}}(z) &= \frac{N_{\frac{n}{2}} \dots N_2 N_1}{D_{\frac{n}{2}} \dots D_2 D_1} X(z) = \left(\prod_{k=1}^{n/2} H_k \right) X(z) = H(z) X(z).
\end{aligned}$$

Implémentation matérielle

En pratique, il est possible d'optimiser la structure des filtres passe-bande en utilisant une configuration « Multiple Feedback » (MFB) (Fig. 6.10). On n'utilise ainsi qu'un seul AOP au lieu de deux si nous avons utilisé un filtre LP et HP séparément.

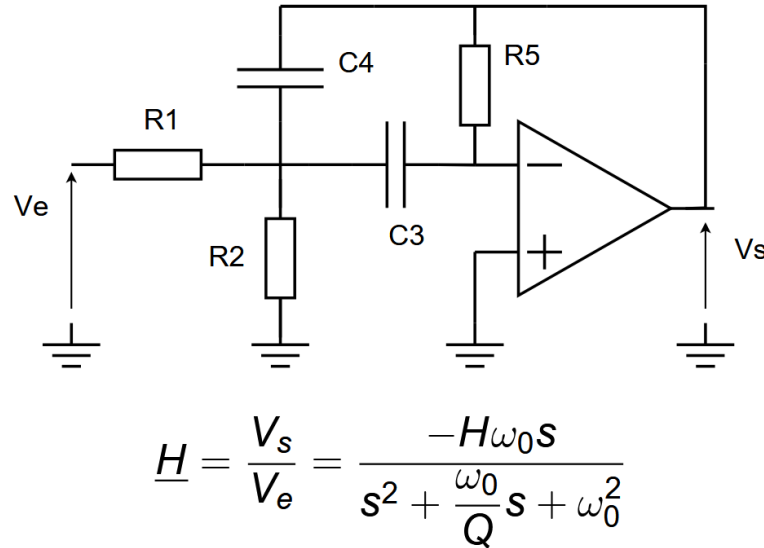


FIGURE 6.10 – Schéma électronique d'un filtre passe bande d'ordre 2 en structure MFB. Les ordres supérieures s'obtiennent par une mise en série de ce filtre.

Simulation du LPPD

Le LPPD (*Low-Pass Peak Detector*) suit la crête instantanée du signal : s'il y a dépassement, la sortie prend la nouvelle valeur, sinon elle décroît exponentiellement.

En discret :

$$y[n] = \begin{cases} x[n] & \text{si } x[n] \geq y[n-1] \\ (1 - \alpha) y[n-1] & \text{sinon} \end{cases}, \quad \alpha = \frac{2\pi f_c}{F_s + 2\pi f_c}.$$

Pour $f_c \ll F_s$, la décroissance suit $e^{-t/\tau}$ avec $\tau = \frac{1}{2\pi f_c}$ ($t = n/F_s$).

6.5 Phases d'apprentissage

Lorsqu'une architecture de type **PrNN** est créée dans le logiciel, l'utilisateur peut lancer un apprentissage complet en appuyant sur le bouton « **Learn** » de l'interface. Cet apprentissage est soumis à des contraintes énergétiques et/ou de performance minimale. L'algorithme final étant complexe, il est découpé en plusieurs phases, décrites ci-dessous.

Dans cette description :

- n représente le nombre de composants utilisés ;
- k le nombre d'échantillons (fichiers audio) ;
- m la taille moyenne des échantillons (nombre de points).

Phase 1 : Initialisation et vérification

Cette phase consiste à (i) formater la structure *PrNN* générée graphiquement, (ii) vérifier sa validité et, en cas d'erreur, générer des messages d'alerte ou d'explication pour guider la correction, puis (iii) effectuer une optimisation numérique de la structure et de l'ordre d'exécution des composants afin de réduire le coût algorithmique des passes suivantes. La complexité algorithmique est de $\mathcal{O}(n)$.

Phase 2 : Acquisition des données

Le *Trigger* est configuré pour se déclencher dès qu'une information étiquetée entre dans le processus de simulation. Un déclenchement périodique est également mis en place afin de capturer des échantillons de bruit dans des proportions homogènes à celles des échantillons d'information. L'ensemble du jeu de données est traité à travers la structure jusqu'à l'enregistrement des tenseurs en entrée du *CRNN*. La complexité algorithmique est de $\mathcal{O}(n) \times k \times m$. Cette étape peut engendrer un coût mémoire et temporel important.

Phase 3 : Sélection des primitives utiles

Cette phase vise à identifier les primitives apportant une réelle valeur à la classification. Le code est exécuté en *Python*. Le réseau de neurones principal est entraîné avec l'ensemble des primitives en entrée, puis un *score d'utilité* est attribué à chacune via la couche *Line Gain*, créée dans le cadre de cette thèse et détaillée en section 7.1.2. Après chaque convergence, la primitive la moins utile est supprimée. Le processus s'arrête lorsque les contraintes énergétiques sont respectées (un nombre N de primitives est atteint) ou que la performance finale chute en dessous du seuil attendu.

Phase 4 : Élagage des primitives

Les primitives jugées inutiles sont supprimées de la structure, à la fois graphiquement et numériquement. L'utilisateur peut visualiser en temps réel la construction du réseau élagué. Le coût algorithmique est négligeable, $\mathcal{O}(1)$.

Phase 5 : Réexécution partielle

Seule la partie analogique reliée au *Trigger* est réexécutée, en utilisant les primitives retenues après élagage. Les tenseurs en entrée sont enregistrés pour les futurs tests d'ap-

prentissage. Comme leur taille est bien inférieure à celle de la phase 2, le coût mémoire et temporel est considérablement réduit.

Phase 6 : Apprentissage du *Trigger*

Un petit réseau de neurones de déclenchement est entraîné à partir des premières informations temporelles des signaux cibles et de signaux de bruit. L'optimisation vise à maximiser le *F1-score* final.

Phase 7 : Simulation finale

Le réseau complet, incluant le *Trigger* précédemment entraîné et fonctionnant à 8 Hz, est exécuté. Les tenseurs en entrée du *CRNN* sont à nouveau enregistrés. Grâce au *buffer* interne du *Multi-Sampler*, aucune donnée n'est perdue : les informations temporelles sont réalignées au début des spectrogrammes, comme le ferait la structure réelle. La complexité est de $\mathcal{O}(n) \times k \times m$. Étant donné l'optimisation du *Trigger* et la réduction du nombre de primitives, le coût mémoire et temporel est inférieur à celui de la phase 2.

Phase 8 : Apprentissage du *CRNN*

Enfin, le *CRNN* est entraîné sur l'ensemble du jeu de données. L'apprentissage s'arrête automatiquement à la convergence, selon des critères statistiques prédéfinis.

À chaque phase, des indicateurs sont présentés en temps réel à l'utilisateur : progression, coût temporel et mémoire, ainsi que les performances atteintes.

Contribution 4 : Algorithmes d'apprentissage des PrNNs

7.1 Apprentissage du bloc d'extraction des primitives (PEX)

7.1.1 Tentative d'optimisation à support libre

La première approche explorée pour concevoir le bloc extracteur de primitives (*PEX*) consistait à fixer un nombre M de filtres passe-bande et à optimiser librement les fréquences de coupure $(\omega_c^{LP}, \omega_c^{HP})_{i=1..M}$ (soit $2M$ paramètres réels, avec la contrainte $\omega_c^{LP} < \omega_c^{HP}$). La non-différentiabilité du problème nous a conduits à tester plusieurs stratégies d'optimisation évolutionnistes reconnues dans la littérature [149–155] :

- Covariance Matrix Adaptation Evolution Strategy (CMA-ES) ;
- Stochastic Ranking Evolution Strategy (SRES) ;
- Unconstrained Evolution Strategy (uES) ;
- Differential Evolution (DE) ;
- Tree-Structured Parzen Estimator (TPE) ;
- Adaptive Simulated Annealing (ASA) ;
- Particle Swarm Optimization (PSO) ;
- Multilevel Coordinate Search (MCS).

Cependant, aucune de ces stratégies n'a été retenue. La raison principale tient au coût d'évaluation prohibitif : en théorie, chaque individu requiert la simulation complète du jeu de données par ses M filtres, l'enregistrement des tenseurs et un test d'apprentissage, ce qui est inenvisageable pour une population très importante et un grand nombre de générations. Par ailleurs, les fortes interactions entre les filtres rendent peu efficaces les opérateurs standards de mutation/croisement : même si un filtre converge aléatoirement vers une région pertinente, l'absence de couverture spectrale conjointe empêche une détection fiable, de sorte que l'évaluation finale demeure faible.

En somme, croiser ou muter des individus n'est réellement efficace qu'à partir du moment où ceux-ci portent des « gènes » utiles en proportion suffisante par rapport à des individus aléatoires. Or, dans notre cas, ces gènes sont difficiles à détecter lorsqu'ils sont rares ; les premières phases de sélection ne peuvent alors pas dépasser l'aléa car elles ne discriminent pas correctement les candidats. Compte tenu de la dimension bornée mais élevée de l'espace de recherche ($2M$ paramètres) et du coût temporel par génération, un support libre s'est avéré empiriquement non performant ; nous avons donc abandonné cette stratégie au profit d'approches plus structurées¹.

7.1.2 Choix d'apprentissage par élagage d'une structure contrainte

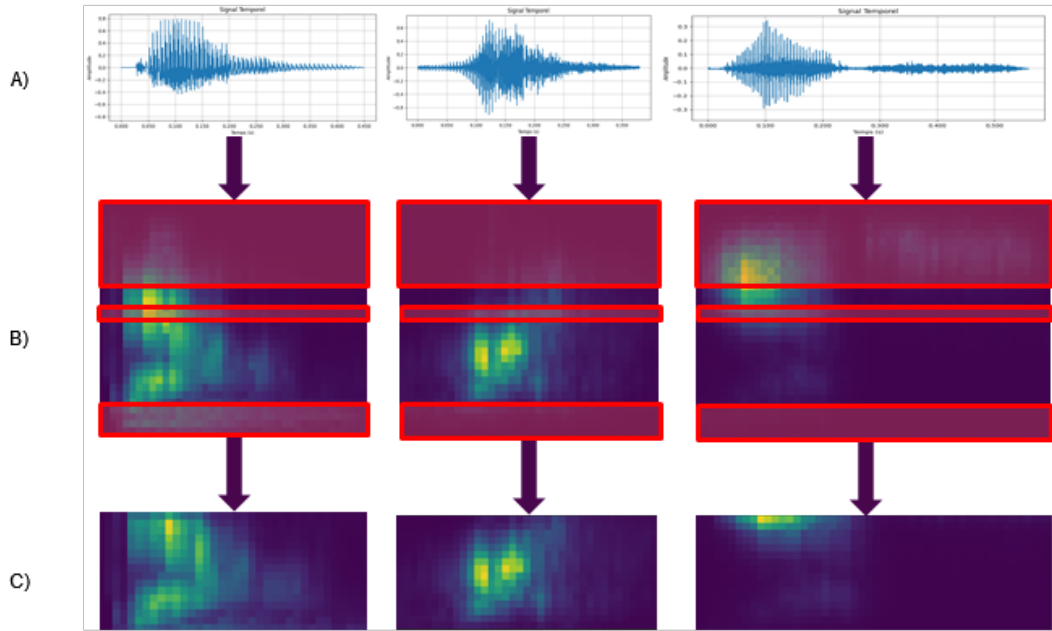


FIGURE 7.1 – Illustration du processus d'optimisation employé : (A) signaux sonores originaux ; (B) spectrogrammes initiaux (les lignes rouges indiquent les primitives jugées les moins pertinentes) ; (C) spectrogrammes après optimisation.

Plutôt que d'essayer d'apprendre les domaines optimaux des filtres, nous allons supposer que la répartition observable chez les êtres vivant est déjà très proche de notre solution. Ainsi, nous figeons les paramètres $(\omega_c^{LP}, \omega_c^{HP})$ de nos filtres selon des échelles prédéfinies (logarithmique, Mel ou linéaire) et, en contrepartie, augmentons la dimension initiale du problème (nombre de filtres). L'objectif n'est plus d'ajuster les domaines fréquentiels, mais de réduire la dimension M du problème, ce qui est bien moins coûteux en calcul. Ce choix sacrifie la possibilité d'atteindre le minimum global, mais assure de trouver des solutions aux performances proches, de manière stable et rapide.

1. Il est pertinent de dire que nous aurions pu nous inspirer des méthodes utilisées dans LEAF [25] (cf. section 1.2.3) plutôt que des stratégies évolutionnistes. Cependant, ce type de simulation de filtre n'a pas été retenu, car jugé trop complexe pour une implémentation analogique. Par ailleurs, à l'instar de notre approche, ces travaux positionnent le point de départ le long d'une échelle de Mel classique.

Pour atteindre notre objectif, nous cherchons à définir une mesure fiable pour quantifier l'utilité de chaque primitive extractrice (c.-à-d. chaque ligne du spectrogramme), afin d'identifier puis de supprimer les caractéristiques les moins pertinentes. Cette tâche est d'autant plus délicate que la pertinence d'une primitive ne dépend pas uniquement d'elle-même, mais aussi des informations qu'elle apporte en combinaison avec les autres. Une illustration de ce processus est présentée en Figure 7.1.

7.1.3 Concept de la couche Line Gain

Afin d'attribuer un score de pertinence à chaque caractéristique d'entrée, nous introduisons une nouvelle couche de réseau de neurones, la « Line Gain » [4, 5]. Placée immédiatement après la couche d'entrée, cette couche applique à chaque caractéristique un poids d'amplification entraînable w_i , jouant de fait, le rôle d'un masque sur les données. Selon la dimensionnalité des données et le contexte, le poids w_i peut s'appliquer à une seule caractéristique d'entrée ou, comme dans notre cas, à un groupe de caractéristiques corrélées par l'acquisition (une ligne du spectrogramme).

L'objectif principal de cette couche est de ramener progressivement certains poids d'amplification vers zéro, afin de diminuer l'influence des caractéristiques les moins pertinentes sur le modèle. Cette dynamique est obtenue en ajoutant une pénalisation L_1 tel que :

$$\text{New Loss} = \text{Original Loss} + \lambda \sum_{i=1}^n |w_i|,$$

où λ est le paramètre de régularisation. Le terme de régularisation $\sum_{i=1}^n |w_i|$, encourage les poids à tendre vers zéro durant l'entraînement. Cependant, du fait de l'hétérogénéité des gradients lors de la backpropagation, tous les poids ne convergent pas uniformément : les poids associés aux entrées peu informatives sont fortement réduits, tandis que ceux correspondant aux caractéristiques essentielles restent stables par compensation des gradients. En pratique, nous utilisons un λ compris entre 0.005 et 0.01.

Malgré sa simplicité, cette méthode induit la formation de matrices creuses, susceptible de dégrader les performances des couches convolutionnelles au fil des époques. Pour palier ce problème, une seule entrée est supprimée à la fois et la couche Line Gain (Lg) est réinitialisée après chaque suppression. Nous introduisons un seuil d'élimination ϵ à partir duquel l'algorithme s'arrête lorsque l'un des poids w_i devient inférieur.

Il peut exister des situations où tous les gains d'amplification se stabilisent au-dessus du seuil ϵ , en particulier lorsque toutes les entrées restantes sont pertinentes. Afin d'éviter un blocage, on introduit également des conditions d'arrêt anticipé (ESC, *Early Stopping Conditions*) avec les seuils de *patience* β et ζ . Si la somme des poids ($\sum_{i=1}^n |w_i|$) et la performance finale du modèle se stabilisent respectivement pendant β et ζ époques, l'apprentissage s'arrête, l'entrée au poids le plus faible est retirée, puis le modèle est réentraîné. Le processus se poursuit jusqu'à satisfaction des critères d'arrêt. Certains des poids sont gardés d'un modèle à l'autre lors de l'entraînement (notamment les noyaux des couches convolutionnelles).

Algorithm 2 Élagage itératif des primitives

Require: Données d'entraînement $(X_{\text{train}}, y_{\text{train}})$, validation $(X_{\text{valid}}, y_{\text{valid}})$, test $(X_{\text{test}}, y_{\text{test}})$; architecture du réseau $\mathcal{M}$; force de régularisation λ ; seuil de suppression ϵ ; paramètres de patience β, ζ ; baisse maximale acceptable de la performance δ ; taille d'entrée cible N_{target}

- 1: Initialiser $\mathcal{M}$ avec le nombre d'entrées N_{in}
- 2: Entraîner $\mathcal{M}$ avec ESC (patience ζ)
- 3: $A_{\text{base}} \leftarrow \text{Eval}(\mathcal{M})$; $A_{\text{cur}} \leftarrow A_{\text{base}}$
- 4: **InputsToDelete** $\leftarrow \emptyset$
- 5: **while** $A_{\text{cur}} \geq A_{\text{base}} - \delta$ **and** $N_{\text{in}} > N_{\text{target}}$ **do**
- 6: Construire $\mathcal{M}'$ à partir de $\mathcal{M}$ en insérant *Line Gain*(λ, N_{in})
- 7: Initialiser $\mathcal{M}'$ avec certains poids de $\mathcal{M}$
- 8: Entraîner $\mathcal{M}'$ avec ESC (ζ) et ESC(*Lg*(λ), ϵ, β)
- 9: $A_{\text{cur}} \leftarrow \text{Eval}(\mathcal{M}')$; $j \leftarrow \text{argmin}([w_i])$
- 10: **if** $A_{\text{cur}} \geq A_{\text{base}} - \delta$ **then**
- 11: **InputsToDelete** $\leftarrow \text{InputsToDelete} \cup \{j\}$
- 12: $N_{\text{in}} \leftarrow N_{\text{in}} - 1$; $\mathcal{M} \leftarrow \mathcal{M}'$
- 13: Retirer l'entrée j de $X_{\text{train}}, X_{\text{valid}}, X_{\text{test}}$ et réduire la dimension d'entrée de $\mathcal{M}$
- 14: **else**
- 15: **break**
- 16: **end if**
- 17: **end while**
- 18: **return** **InputsToDelete**, $\mathcal{M}$

7.1.4 Considération sur l'absorption des bandes supprimées

Nous avons envisagé de maintenir une forme d'*absorption spectrale* des bandes supprimées par les bandes adjacentes. À la fin de chaque cycle, le domaine de la bande supprimée est subdivisé et l'on élargit le domaine des bandes adjacentes selon K variantes, créant ainsi K versions différentes du modèle. Cette idée produit une solution hybride entre une exploration génétique et l'élagage sous contrainte, permettant de s'approcher de la solution optimale au prix d'un facteur K sur le coût temporel (ce qui peut être jugé acceptable).

Du point de vue strictement des performances, cette stratégie ne peut qu'être bénéfique; toutefois, des contraintes matérielles s'y opposent. La possibilité de moduler les filtres suppose soit une fabrication sur mesure pour chaque modèle d'IA, soit l'emploi de filtres reconfigurables (résistances variables). La première option n'est pas économiquement intéressante; la seconde est plus énergivore et complexifie la conception, ce qui, à budget énergétique constant, réduit le nombre de primitives déployables. Dans ces conditions, le gain attendu de l'absorption du domaine spectral ne compense pas les inconvénients associés.

7.1.5 Autres exemples d'application

Le problème du choix des entrées d'un modèle dépasse largement le cas étudié ici. Dans de nombreuses situations, on dispose d'un nombre de variables d'entrée supérieur à ce qui est nécessaire pour analyser ou modéliser un phénomène. L'utilisation d'une couche *Line Gain* constitue alors un moyen simple et efficace de distinguer les informations pertinentes des caractéristiques superflues.

Pour l'illustrer, nous menons une expérience inspirée de *TensorFlow Playground* [156]. Nous considérons deux classes de points (rouge et bleu) dispersés dans le plan selon une règle de génération fixe. À chaque point sont associées plusieurs « primitives » (caractéristiques candidates). L'objectif est de modéliser la règle de séparation à l'aide d'un réseau de neurones tout en minimisant le nombre de primitives retenues.

Nous réalisons l'expérience pour quatre types de règles : *Cercle*, *XOR*, *Spirale* et *Gaussien*. Les primitives initialement présentées au réseau sont x , y , x^2 , y^2 , xy , $\sin(x)$ et $\sin(y)$. Notre protocole expérimental suit celui décrit précédemment ; l'architecture détaillée du modèle et les hyperparamètres utilisés lors des phases d'apprentissage et d'élagage sont reportés dans les Tableaux 7.1 et 7.2.

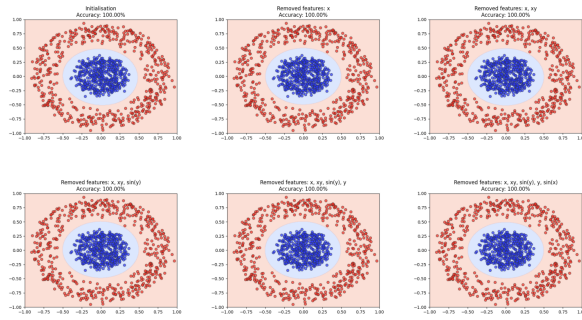
TABLE 7.1 – Architecture du réseau de neurones utilisée dans l'expérimentation

Couche	Unités	Activation	Régularisation
Input	7	–	–
LineGain	7	–	L1(0.007)
Dense	5	tanh	L2(0.001)
Dense	5	tanh	L2(0.001)
Dense	4	tanh	L2(0.001)
Dense	2	tanh	L2(0.001)
Dense	1	sigmoid	–

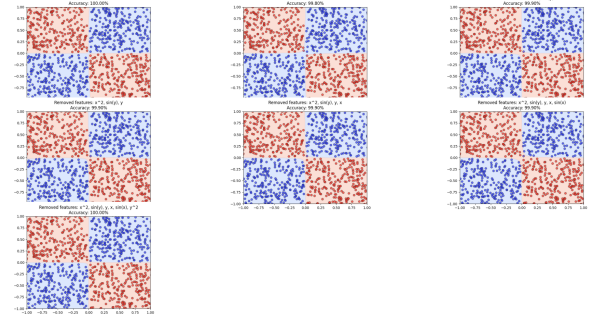
TABLE 7.2 – Hyperparamètres utilisés lors de l'apprentissage et de l'élagage

Paramètre	Valeur
Optimiseur	Adam
Fonction de perte	Binary Crossentropy
Nombre maximal d'époques ζ	500
Intensité λ	0.007
Seuil d'élagage ϵ	0.001

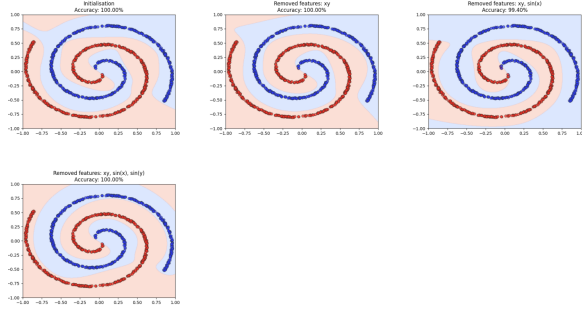
Les résultats de l'élagage, illustrés en Figure 7.2, montrent que l'importance des entrées varie selon la règle considérée. Bien qu'elles soient toutes exploitées initialement par le modèle, une redondance latente subsiste ; notre méthode permet de l'identifier puis de l'éliminer.



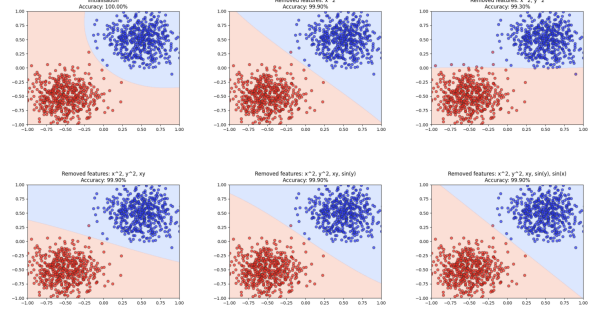
(a) Répartition en cercle. Ordre de suppression : $[x, xy, \sin(y), y, \sin(x)]$. Primitives utilisées : (x^2, y^2)



(b) Répartition XOR. Ordre de suppression : $[x^2, \sin(y), y, x, \sin(x), y^2]$. Primitives utilisées : (xy)



(c) Répartition en spirale. Ordre de suppression : $[xy, \sin(x), \sin(y)]$. Primitives utilisées : (x, y, x^2, y^2)



(d) Répartition en deux gaussiennes. Ordre de suppression : $[x^2, y^2, xy, \sin(y), \sin(x)]$. Primitives utilisées : (x, y)

FIGURE 7.2 – Résultat de classification du réseau de neurones après la suppression successive de chaque primitive.

7.2 Apprentissage du Trigger

7.2.1 Idée générale

L'objectif du bloc *Trigger* est d'exploiter les primitives extraites pour décider si l'information courante mérite d'être traitée, de façon à ne déclencher l'échantillonnage et le réseau de neurones qu'un nombre minimal de fois. Pour ce faire, un petit réseau de neurones s'exécute à 8 Hz sur des valeurs issues des signaux analogiques et leurs versions filtrées en passe-bas. En parallèle, les primitives sont échantillonnées à 100 Hz dans un tampon interne de 16 échantillons (soit 0.16 s). Lorsqu'une acquisition est ordonnée (à 100 Hz également), le contenu de ce tampon est préfixé au tenseur d'entrée final, garantissant la conservation du contexte temporel précédant le déclenchement. Afin de garantir un apprentissage pertinent, nous utilisons des séquences temporelles suffisamment longues pour que les filtres analogiques puissent converger.

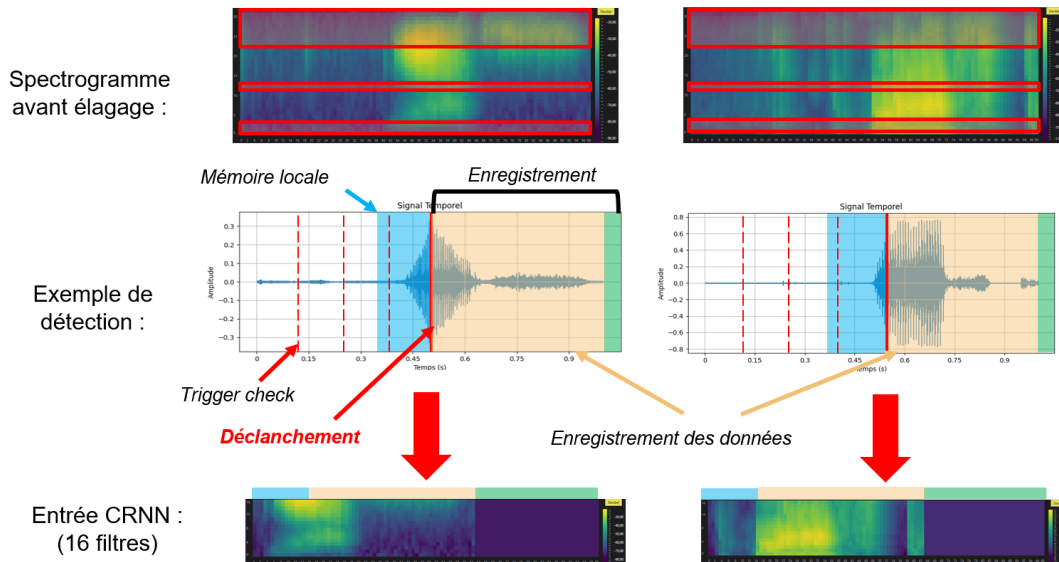


FIGURE 7.3 – Visualisation temporelle du processus d'échantillonnage. (A) Élagage des primitives d'origine (en rouge). (B) Visualisation temporelle des inférences du réseau de déclenchement sur le signal audio. (C) Résultat final. Le tenseur est complété par des zéros si l'enregistrement se termine après la fin du fichier audio.

7.2.2 Modèles envisagés

Nous avons exploré plusieurs architectures de réseau de neurones, différant par la représentation échantillonnées en entrée et le traitement du contexte temporel.

(1) *Entrée concaténée des primitives et de leur filtrage en passe-bas.* Le réseau reçoit, à l'instant d'évaluation, un tenseur constitué des N valeurs des primitives et des N valeurs de leur filtrage passe-bas, soit une entrée de taille $2N$. L'architecture est un perceptron multicouche comprenant quatre couches denses (16–8–8–1).

(2) *Entrée normalisée par rapport au passe-bas.* Les valeurs des primitives sont divisées par leurs versions filtrées², fournissant une entrée de taille N . Cette transformation vise à décorréler les bandes fréquentielles et à réduire la sensibilité aux variations d’amplitude.

(3-4) *Variantes avec contexte temporel empilé.* Pour capturer la dynamique, nous concaténons huit fenêtres successives échantillonnées à 50 Hz (valeurs lues dans les tampons internes), produisant des « images » de taille $8 \times 2N$ pour la variante (1) et $8 \times N$ pour la variante (2). Les réseaux correspondants sont composés d’une couche GRU afin d’analyser plus précisément l’évolution de la structures spatiales.

Constitution du jeu de données.

Pour constituer notre jeu de données, le simulateur n’exécute que la partie analogique du PrNN. Lorsqu’un tenseur retenu (dans le cas des variantes (3) et (4), il s’agit des dernières informations acquises) correspond au début d’une vocalise, l’échantillon est annoté *positif* et enregistré. Aucun autre échantillon n’est alors conservé pendant la durée théorique d’enregistrement de l’ADC principal. À l’issue de cette période, si le tenseur suivant tombe à nouveau dans un intervalle actif de vocalise, il est enregistré comme *positif* et un nouveau cycle d’attente commence.

Parallèlement, des tenseurs correspondant à des périodes de bruit sont prélevés périodiquement. Tous ne sont toutefois pas conservés, de manière à maintenir un rapport de classes équilibré et à éviter un biais d’entraînement au détriment des exemples rares.

Les performances des modèles entraînés sur ces jeux de données, ainsi que l’impact des différentes méthodologies de traitement, seront présentées dans la section suivante.

2. Cette pratique est une version *low-cost* de l’idée proposée par la méthode *PCEN* (*Per-channel energy normalization*)

Partie III

Résultats & Perspectives

Datasets utilisés

Afin de valider expérimentalement notre approche, nous considérons trois tâches de reconnaissance acoustique distinctes : (I) la détection de mots-clés humains (*Keyword Spotting*, KWS), (II) l'identification d'espèces d'oiseaux et (III) la présence de chauves-souris. Ces choix couvrent des domaines sonores et des environnements d'acquisition très différents, offrant un banc d'essai représentatif. En outre, ces domaines disposent d'une littérature psychoacoustique riche, ce qui nous permet de vérifier que l'évolution des PrNNs demeure cohérente avec les connaissances biologiques établies.

8.1 Dataset humain

Pour la détection humaine, nous utilisons le jeu de données *Google Speech Commands Dataset (GSCD)* [70]. Plus précisément, nous utilisons un sous-ensemble sélectionné¹ contenant huit mots-clés distincts : *Yes*, *No*, *Left*, *Right*, *Up*, *Down*, *Go* et *Stop*. Chaque classe est représentée par 1 000 échantillons audio d'environ une seconde, aboutissant à un jeu de données équilibré de 8 000 échantillons.

8.2 Dataset d'oiseaux

Les jeux de données open-source recensés pour les oiseaux présentent plusieurs limitations majeures : (i) les annotations portent généralement sur l'intégralité du fichier, et non au niveau de la vocalisation ; (ii) plusieurs espèces coexistent fréquemment dans un même enregistrement et certaines ne sont pas étiquetées, en particulier lorsqu'elles restent en arrière-plan ; (iii) aucun contrôle ni indicateur du rapport signal/bruit (SNR) n'est fourni, ce qui ne permet pas d'évaluer la robustesse du modèle final.

En réponse à ces limitations, nous avons construit et mis à disposition notre propre dataset [6].

1. http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands.zip

8.2.1 Création du jeu de données

Le jeu de données utilisé dans cette étude a été construit à partir d'enregistrements issus de la base *Xeno-Canto* [14]. Nous avons sélectionné 10 espèces d'oiseaux cohabitant dans des régions similaires en Europe et présentant des patrons d'activité nocturne comparables.

Les enregistrements *Xeno-Canto* sont notés avec un score de qualité (de A à E) qui est attribué subjectivement par le dépositaire de l'enregistrements. Pour garantir la fiabilité des données initiales, seuls les enregistrements notés A ou B ont été retenus. Comme les annotations s'appliquent à l'intégralité des fichiers, chaque fichier a été ré-annoté manuellement avec une précision à la milliseconde. Chaque espèce est représentée par environ 45 minutes d'activité vocale.

TABLE 8.1 – Composition du jeu de données d'oiseaux extrait du dépôt Xeno-Canto

Nom	Label	Durée audio	Temps actif
Turdus merula	TuMe	65 min	26 min
Erithacus rubecula	ErRu	131 min	43 min
Cyanistes caeruleus	CyCa	141 min	41 min
Fringilla coelebs	FrCo	153 min	45 min
Carduelis carduelis	CaCa	79 min	48 min
Corvus corone	CoCo	134 min	42 min
Columba palumbus	CoPa	112 min	46 min
Hirundo rustica	HiRu	82 min	55 min
Alauda arvensis	AlAr	70 min	67 min
Luscinia megarhynchos	LuMe	100 min	44 min
Total		19h28min	7h47min

Afin d'enrichir le corpus initial, nous avons construit un second jeu de données à partir des sons environnementaux de la base *ESC-50* [157]. Vingt-trois classes ont été retenues (Tableau 8.2), chacune comprenant 40 extraits audio uniques de 5 s. Tous les échantillons ont été réannotés manuellement afin de délimiter précisément leurs bornes temporelles et spectrales. Étant donné que la distribution de ces bruits varie selon les écosystèmes, une estimation statistique a été menée et ajustée aux conditions climatiques locales, de manière à préserver la pertinence écologique.

En complément, nous avons intégré des bruits climatiques de longue durée (pluie, vent) issus de la base *Pixabay*. Ces enregistrements sont fournis à différents niveaux d'intensité ; leurs fréquences d'occurrences et leurs dynamiques d'intensités sur une année ont également été prises en compte afin de garantir la plausibilité écologique.

Dans une seconde étape, tous les fichiers ont été filtrés temporellement et spectralement afin d'éliminer les erreurs d'annotation et les artefacts de bruit non contrôlés (cf. problématique (ii)). Le niveau RMS *actif* de chaque échantillon a été normalisé à 0.05, correspondant typiquement au niveau d'un chant d'oiseau enregistré à 2 m. Une rampe de gain linéaire de 0.5 seconde a été appliquée au début et à la fin de chaque extrait pour lisser les transitions. Pour assurer la compatibilité lors du mélange, tous les fichiers audio ont été rééchantillonnés à 48 kHz.

TABLE 8.2 – Classes des bruits utilisés et leurs probabilités de sélection conditionnelle. Le taux de recouvrement moyen de ces événements est fixé à 65 %.

¹Des probabilités globales d’occurrence de 15 % (pluie) et 30 % (vent), avec variation stochastique de l’intensité, sont retenues ; les événements de vent sont cumulatifs.

Nom	Type	Prob. temps sec ¹	Prob. pluie ¹
insects	animal	0.18	—
crickets	animal	0.16	—
frog	animal	0.16	0.10
dog	animal	0.08	—
cat	animal	0.03	0.02
cow	animal	0.03	0.02
hen	animal	0.03	0.02
pig	animal	0.02	—
rooster	animal	0.02	—
sheep	animal	0.02	—
footsteps	urbain	0.10	0.05
church_bells	urbain	0.03	0.03
engine	urbain	0.03	0.03
car_horn	urbain	0.01	0.02
chainsaw	urbain	0.01	—
train	urbain	0.02	0.02
airplane	urbain	0.01	0.01
fireworks	urbain	0.01	—
helicopter	urbain	0.01	0.01
sneezing	urbain	0.01	0.01
sea_waves	naturel	0.02	0.02
crackling_fire	naturel	0.01	—
thunderstorm	naturel	—	0.32
silence	—	—	0.32

Lors de la phase de mélange, un rapport signal/bruit (SNR) moyen de 1 a été adopté comme référence. Pour chaque segment audio, un contexte climatique est tiré au sort selon la distribution pondérée. En fonction de l’environnement sélectionné, une décision probabiliste (65 %) est prise toutes les 5 secondes pour déterminer l’insertion éventuelle d’un événement de bruit. Le cas échéant, l’événement est positionné dans la fenêtre correspondante, avec un décalage temporel aléatoire allant jusqu’à ± 2 secondes.

Afin de préserver la redondance écologique², un maximum de trois classes de bruit distinctes est autorisé par fichier audio. Une fois ce seuil atteint, les probabilités de sélection sont ajustées de manière à privilégier soit les classes déjà choisies, soit le silence. Pour simuler des variations de distance d’enregistrement, les fichiers sont amplifiés et combinés pour atteindre le SNR cible, puis soumis à une atténuation aléatoire supplémentaire comprise entre 0 et -20 dB.

Un aperçu du processus complet de la construction est présenté en Figure 8.1.

2. Un facteur exploité plus tard par le module *Sensitivity Adaptator*.

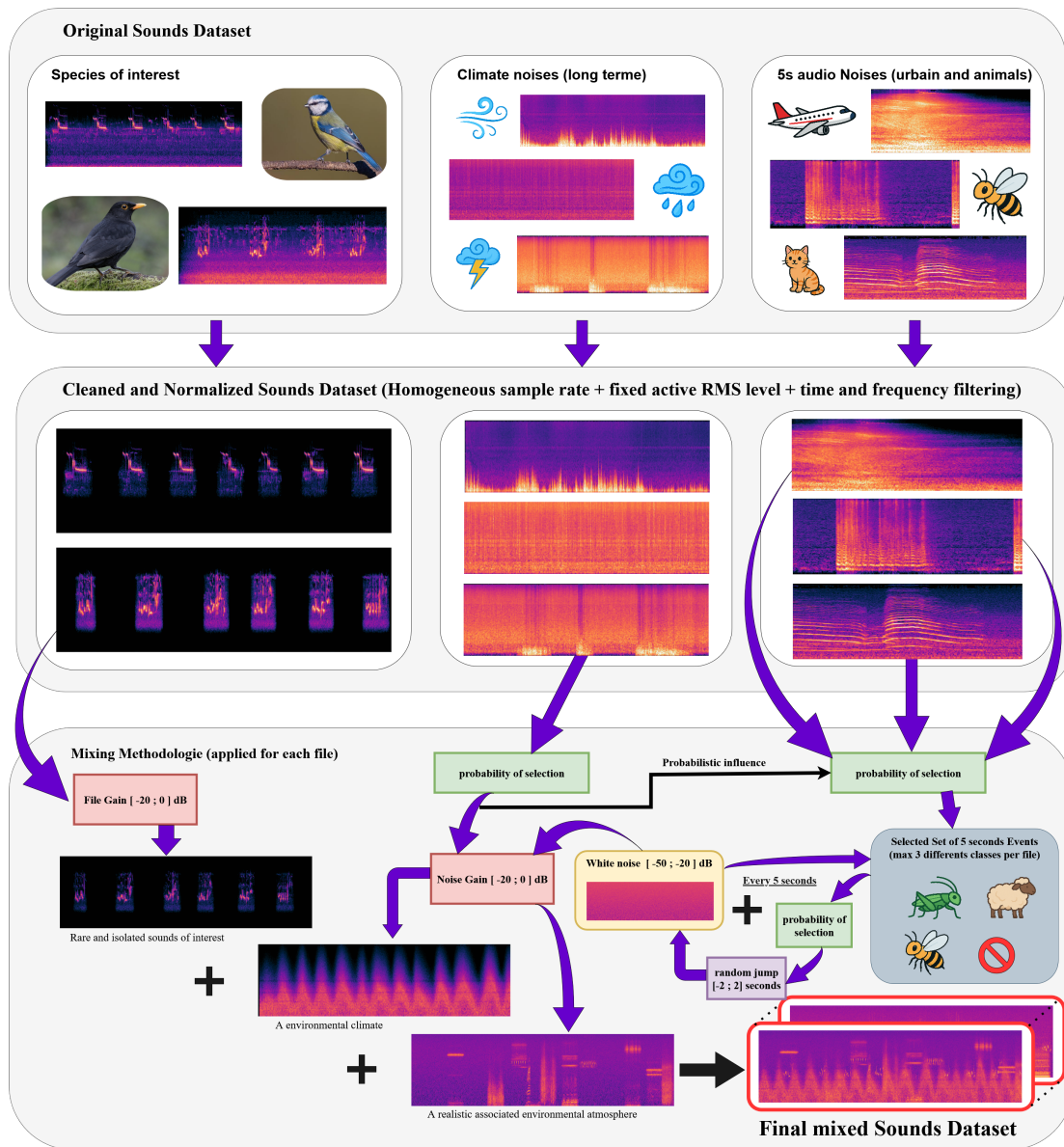


FIGURE 8.1 – Vue d’ensemble du processus de construction et d’augmentation du jeu de données d’oiseaux.

8.2.2 Pertinence de la méthodologie

Ce pipeline a abouti à deux jeux de données d'environ 20 heures chacun³. Le premier, noté $\mathcal{D}_{\text{original}}$, contient les enregistrements d'origine de haute qualité (score A et B). Le second, noté $\mathcal{D}_{\text{augmented}}$, reprend les mêmes enregistrements nettoyés et enrichis de bruits environnementaux, avec un SNR de 1 (0 dB). Pour évaluer la pertinence de notre enrichissement par rapport aux conditions réelles, un troisième jeu de test, $\mathcal{D}_{\text{real}}$, a été constitué à partir d'enregistrements de qualité moyenne (score C).

Cette évaluation se fait avec le modèle BirdNet v2.4, une référence largement reconnue pour la reconnaissance des sons d'oiseaux et réutilisée par la suite. Durant l'entraînement, chaque fichier audio est découpé en fenêtres de 3 secondes. Un segment est annoté positif pour toute classe présente plus de 0.5 seconde ; sinon, il est considéré comme du bruit de fond. Les jeux $\mathcal{D}_{\text{original}}$ et $\mathcal{D}_{\text{augmented}}$ sont chacun divisés en ensembles d'entraînement, de validation et de test selon une répartition 80/10/10 %.

Pour évaluer l'efficacité de l'approche sur des jeux plus réduits, une seconde expérimentation a été menée. Dans ce protocole, seulement 20 % de $\mathcal{D}_{\text{original}}$ (environ 4 heures d'audio) a été utilisé pour l'entraînement. La même procédure d'augmentation que pour $\mathcal{D}_{\text{augmented}}$ a été appliquée, mais répétée cinq fois par fichier, reportant la durée totale à 20 heures. Les jeux résultants sont notés $\mathcal{D}_{\text{original-red}}$ et $\mathcal{D}_{\text{augmented-ext}}$. Les trois jeux de test de la première expérimentation ont été réutilisés pour l'évaluation.

Les résultats du Tableau 8.3 montrent que le modèle entraîné sur le jeu d'origine $\mathcal{D}_{\text{original}}$, puis évalué sur son jeu de test associé obtient les meilleures performances, mais ce score s'explique en grande partie car la qualité audio est élevée et le contexte acoustique restreint. Lorsque le modèle est confronté à des enregistrements de terrain plus réalistes et bruités, ses performances chutent nettement, soulignant une robustesse limitée hors conditions contrôlées (cf. problématique (iii)).

À l'inverse, le modèle entraîné sur $\mathcal{D}_{\text{augmented}}$ se montre plus robuste. Le réseau maintient des capacités de détection stables entre les données synthétiques et les données réelles. Ce résultat prouve la pertinence de notre méthodologie pour créer le jeu de données et montre clairement le biais de performance qu'il peut y avoir par rapport à la réalité sinon.

Cet écart de performance apparaît encore plus clairement dans le Tableau 8.4, où l'entraînement est réalisé sur un jeu réduit. Si le premier modèle ne semble pas moins performant sur les jeux originaux et augmentés, il s'avère en pratique moins efficace. À l'opposé, le modèle entraîné avec augmentation ne subit qu'une légère dégradation par rapport à l'expérience précédente.

3. <https://github.com/AdrienDeverin/Birds-10-Dataset>

TABLE 8.3 – Performances du modèle (PR-AUC par classe, PR-AUC moyenne et F1-score moyen) selon les combinaisons de jeux d’entraînement et de test.

Jeu d’entraîn.	Jeu de test	TuMe	CoCo	CyCa	AlAr	CoPa	CaCa	FrCo	HiRu	LuMe	ErRu	Moy.	F1 moy.
$\mathcal{D}_{\text{original}}$	$\mathcal{D}_{\text{original}}$	0.96	0.95	0.97	0.99	0.94	0.98	0.94	0.97	0.97	0.92	0.96	0.95
$\mathcal{D}_{\text{original}}$	$\mathcal{D}_{\text{augmented}}$	0.69	0.83	0.81	0.88	0.60	0.84	0.67	0.74	0.70	0.76	0.75	0.64
$\mathcal{D}_{\text{original}}$	$\mathcal{D}_{\text{real}}$	0.67	0.81	0.79	0.83	0.57	0.83	0.64	0.70	0.69	0.73	0.72	0.59
$\mathcal{D}_{\text{augmented}}$	$\mathcal{D}_{\text{original}}$	0.92	0.94	0.97	0.98	0.85	0.97	0.90	0.96	0.97	0.91	0.94	0.90
$\mathcal{D}_{\text{augmented}}$	$\mathcal{D}_{\text{augmented}}$	0.91	0.92	0.96	0.96	0.64	0.93	0.91	0.92	0.91	0.90	0.90	0.86
$\mathcal{D}_{\text{augmented}}$	$\mathcal{D}_{\text{real}}$	0.90	0.92	0.93	0.95	0.65	0.92	0.89	0.90	0.90	0.90	0.89	0.85

TABLE 8.4 – Modèle entraîné sur un jeu réduit. Performance (PR-AUC par classe, PR-AUC moyenne et F1-score moyen) évaluée sur différents jeux de test.

Jeu d’entraîn.	Jeu de test	TuMe	CoCo	CyCa	AlAr	CoPa	CaCa	FrCo	HiRu	LuMe	ErRu	Moy.	F1 moy.
$\mathcal{D}_{\text{original-red}}$	$\mathcal{D}_{\text{original}}$	0.95	0.93	0.97	0.99	0.92	0.97	0.92	0.98	0.96	0.91	0.95	0.93
$\mathcal{D}_{\text{original-red}}$	$\mathcal{D}_{\text{augmented}}$	0.69	0.84	0.87	0.88	0.61	0.88	0.71	0.77	0.72	0.77	0.77	0.67
$\mathcal{D}_{\text{original-red}}$	$\mathcal{D}_{\text{real}}$	0.60	0.80	0.75	0.78	0.55	0.79	0.61	0.67	0.63	0.70	0.69	0.55
$\mathcal{D}_{\text{augmented-ext}}$	$\mathcal{D}_{\text{original}}$	0.91	0.93	0.97	0.97	0.84	0.96	0.91	0.92	0.95	0.90	0.93	0.89
$\mathcal{D}_{\text{augmented-ext}}$	$\mathcal{D}_{\text{augmented}}$	0.90	0.92	0.95	0.95	0.65	0.91	0.90	0.90	0.91	0.89	0.89	0.85
$\mathcal{D}_{\text{augmented-ext}}$	$\mathcal{D}_{\text{real}}$	0.89	0.92	0.94	0.93	0.66	0.91	0.87	0.90	0.89	0.90	0.88	0.85

L'efficacité de la méthodologie proposée a également été évaluée sous le contrôle de plusieurs SNR, comme illustré en Figure 8.2.

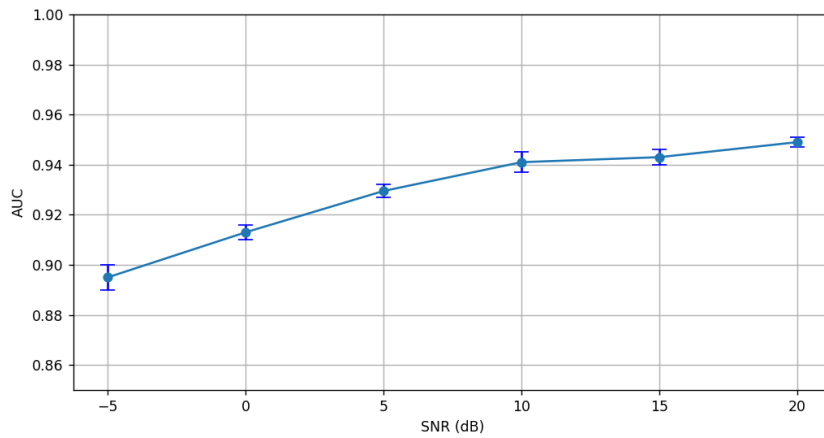


FIGURE 8.2 – Évaluation des performances du modèle BirdNet entraîné sur $\mathcal{D}_{\text{augmented}}$, créée avec différents rapport signal-bruit (SNR) et sans atténuation globale.

8.3 Dataset de chauves-souris

Les enregistrements de chauves-souris utilisés dans cette étude proviennent du dépôt *ChiroVox* et correspondent à l'espèce *Scotophilus kuhlii*, couramment observée en Asie. La même méthodologie de construction du jeu de données que celle décrite pour les oiseaux a été appliquée. Pour cette tâche, l'échantillonnage des primitives se fait à 300 Hz et sur une durée de 0.2 seconde, soit 60 échantillons.



FIGURE 8.3 – Image d'une *Scotophilus kuhlii*, ou « chauve-souris jaune naine d'Asie ».

Résultat de l'apprentissage de l'extraction des primitives

9.1 Cas d'application 1 : Keyword Spotting (KWS)

9.1.1 Choix de l'ordre des filtres passe-bande

Comme détaillé dans le chapitre 5, l'ordre des filtres passe-bande utilisés pour les primitives influence à la fois la qualité de détection et la consommation énergétique. Pour établir notre décision, nous avons comparé les performances à partir des Mel-spectrogrammes obtenus avec des filtres d'ordre 2, 4, 6 et par transformée de Fourier (FFT)¹. La décision annoncée précédemment de maintenir un ordre 4 pour l'interface matérielle, découle des résultats fournis dans le Tableau 9.1. Comme on peut le voir, un ordre supérieur n'améliore pas de manière significative la qualité de détection pour cette tâche.

9.1.2 Évolution de la précision en fonction du nombre de filtre

La figure 9.1 présente l'évolution de la précision du modèle *EcoPhonic* en fonction du nombre de filtres en entrée. Au cours du processus d'élagage, deux phases distinctes apparaissent :

- *Les Phases d'optimisation*, qui correspondent à une période durant laquelle la précision baisse très légèrement mais est compensée par des gains énergétiques substantiels. Aucune information critique n'est perdue ; seules les redondances et des informations faiblement utiles sont supprimées.
- *Les Phases d'oubli*, qui surviennent lorsqu'un filtre retiré contribue de façon déterminante à une discrimination phonétique.

1. Ici, les spectres évalués ont une résolution de 64×100 , à l'exception du modèle *Whisper*, qui requiert des entrées de résolution 80×100 et traite cette entrée avec 29 secondes supplémentaires (zero-padding), d'où les valeurs très élevées. Dans le cas de *Whisper*, l'encodeur est utilisé tel quel ; seul le classificateur (deux couches denses) est entraîné.

Modèle	MMAC	MB	2 ^e ordre (%)	4 ^e ordre (%)	6 ^e ordre (%)	FFT (%)
Whisper (base) [19]	$\approx 50\,000.0$	≈ 300.00	86.1	92.5	93.7	97.0
YOLO_v5s [20]	90.0	36.60	90.9	93.8	94.7	95.1
Edge_CRNN (1.0x) [75]	16.4	2.65	90.0	92.1	92.3	94.7
EcoPhonic (nous)	5.1	0.18	90.8	94.4	94.9	95.0

TABLE 9.1 – Comparaison des performances et des ressources requises selon différentes techniques d’extraction de spectrogrammes. Les précisions sont reportées pour des filtres analogiques d’ordres 2, 4 et 6, échantillonnés à 100 Hz, ainsi que pour une FFT numérique. L’usage de la mémoire *flash* est exprimé en *MB* (4 octets/valeur) et les *MMAC* correspondent au nombre d’opérations multiplication–accumulation par inférence (en millions).

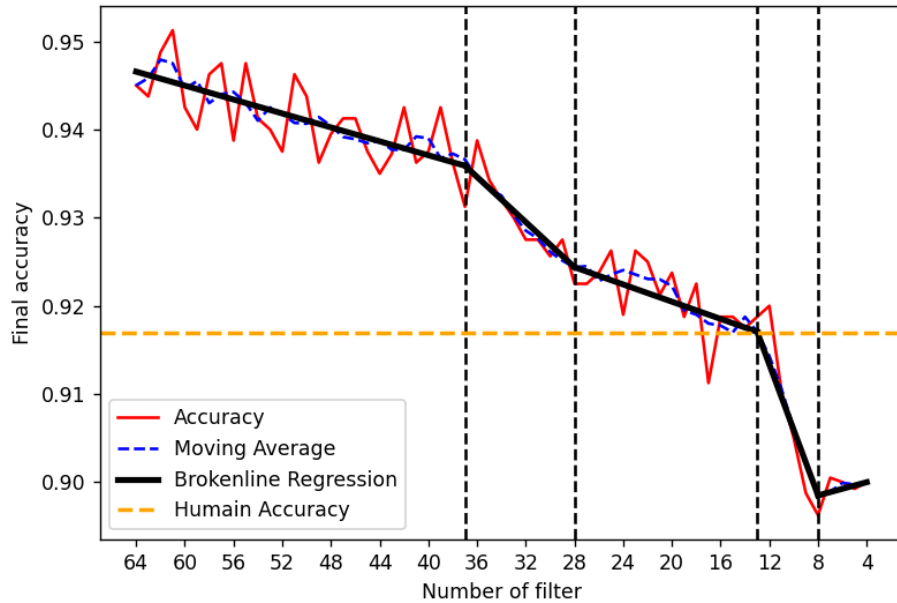


FIGURE 9.1 – Précision finale du réseau *EcoPhonic* en fonction du nombre de filtres (de 64 à 4) pour une tâche de KWS avec 8 classes. En rouge : valeurs ponctuelles ; en bleu : moyenne glissante ; en noir : tendance (régressions linéaires par morceaux).

Ces deux phases peuvent alterner à plusieurs reprises, en raison de la diversité phonétique du jeu de données et de la distribution inégale de l’information partagée entre les filtres. De ce fait, la suppression d’un seul filtre au sein d’un sous-groupe peut rendre l’ensemble de ce sous-groupe inopérant (phénomène observé entre la 27^e et la 36^e suppression). Il est néanmoins remarquable qu’avec seulement quatre primitives, la précision ne soit inférieure que de 5 % au maximum atteint par le modèle.

Par ailleurs, contrairement à l’intuition, la précision humaine sur ce jeu de données n’est pas parfaite : elle est estimée en moyenne à 91.8 % (repère orange). Rapportée à nos

résultats, cette valeur se situe juste avant la phase finale d’oubli, aux alentours de 13 primitives, en cohérence avec nos estimations énergétiques.

9.1.3 Lien biologique

Le tableau 9.2 fournit les bandes de fréquence exactes des primitives initiales ainsi que leur ordre de suppression.

TABLE 9.2 – Fréquence de coupure des 64 bandes de filtres initiales (échelle Mel) pour la tâche KWS. L’ordre de suppression du processus d’élagage est le suivant :

64, 1, 2, 3, 45, 63, 62, 61, 50, 52, 59, 58, 57, 56, 51, 43, 4, 7, 8, 60, 53, 9, 10, 11, 55, 49, 24, 22, 29, 30, 13, 20, 14, 15, 16, 48, 17, 18, 19, 47, 21, 46, 23, 44, 31, 42, 32, 41, 33, 40, 27, 34, 35, 39, 36, 38, 28, 6, 25, 37.

Numéro	Bande Fréq. (Hz)	Numéro	Bande Fréq. (Hz)
1	50 – 79	33	1855 – 1954
2	79 – 110	34	1954 – 2057
3	110 – 141	35	2057 – 2164
4	141 – 174	36	2164 – 2276
5	174 – 209	37	2276 – 2394
6	209 – 244	38	2394 – 2515
7	244 – 281	39	2515 – 2640
8	281 – 319	40	2640 – 2770
9	319 – 358	41	2770 – 2905
10	358 – 400	42	2905 – 3045
11	400 – 443	43	3045 – 3194
12	443 – 488	44	3194 – 3345
13	488 – 534	45	3345 – 3503
14	534 – 582	46	3503 – 3666
15	582 – 632	47	3666 – 3836
16	632 – 684	48	3836 – 4012
17	684 – 738	49	4012 – 4200
18	738 – 795	50	4200 – 4390
19	795 – 853	51	4390 – 4588
20	853 – 913	52	4588 – 4794
21	913 – 976	53	4794 – 5008
22	976 – 1041	54	5008 – 5230
23	1041 – 1109	55	5230 – 5460
24	1109 – 1181	56	5460 – 5706
25	1181 – 1254	57	5706 – 5955
26	1254 – 1330	58	5955 – 6213
27	1330 – 1409	59	6213 – 6482
28	1409 – 1491	60	6482 – 6762
29	1491 – 1576	61	6762 – 7052
30	1576 – 1667	62	7052 – 7361
31	1667 – 1759	63	7361 – 7674
32	1759 – 1855	64	7674 – 7999

On observe que la sélection des filtres au cours de l’élagage semble davantage guidée par des principes de perception auditive humaine que par de simples heuristiques empiriques.

Durant la première phase, le domaine spectral est progressivement resserré entre 400 et 4000 Hz, tout en préservant quelques bandes spécifiques : les filtres **5 et 6 (170–240 Hz)** ainsi que le filtre **54 (5000–5230 Hz)**. Cette convergence vers ce domaine spectral concorde avec les connaissances psychoacoustiques [158] : c’est dans cette plage que se concentre l’essentiel de l’information de la parole, notamment les trois premiers formants (F_1 , F_2 , F_3), déterminants pour la distinction des phonèmes [159].

À mesure que l’élagage progresse, les filtres couvrant la bande médiane (500–1500 Hz) sont retirés. Bien qu’ils contiennent une information phonétique, les résultats empiriques montrent que leur suppression, au regard de l’information restante, entraîne une dégradation minimale des performances. Cela correspond à la première *phase d’oubli* mise en évidence dans la figure 9.1.

Par la suite, l’algorithme se met à élaguer de manière alternée les fréquences dans les bandes 1500-2000 Hz et 3000-4000 Hz, signe d’une stratégie de destruction minimale. Lorsque le processus atteint les 13 derniers filtres, seules quelques régions ciblées restent actives : 170–250 Hz, 400-500 Hz, 1250–1500 Hz, 2000–3000 Hz et 5000–5200 Hz.

Une réduction plus poussée amène à supprimer les fréquences comprises entre 2000 et 3000 Hz, ce qui correspond à la partie haute du troisième formant (F_3) et la dernière *phase d’oubli*. Là encore, ce choix de suppression s’aligne avec les résultats de la littérature, qui indiquent que l’intelligibilité de la parole repose davantage sur les bandes basses et médianes [160].

Les huit derniers filtres se répartissent sur les zones clés du spectre :

- **Filtres 5 et 6 (170–240 Hz)** : captent des composantes basses essentielles à la distinction des fondamentales.
- **Filtre 12 (440–490 Hz)** : correspond aux basses fréquences du premier formant (F_1), importantes pour la différenciation des voyelles [161].
- **Filtres 25, 26 et 28 (1180–1500 Hz)** : couvrent la zone du second formant (F_2), cruciale pour certaines consonnes et la qualité des voyelles.
- **Filtre 37 (2270–2400 Hz)** : proche du troisième formant (F_3), utile pour distinguer des phonèmes acoustiquement voisins [159].
- **Filtre 54 (5008–5230 Hz)** : capte des consonnes à haute fréquence (/s/ dans *yes* et /f/ dans *left*).

9.1.4 Impact de la répartition initiale

Nous avons étudié dans quelle mesure les conditions initiales influencent les performances finales. À cette fin, trois séries de simulations ont été menées en faisant varier (i) le nombre de primitives initiales (64, 32 et 24) et (ii) l’échelle auditive utilisée pour leur répartition (Mel et Bark). Dans l’ensemble des configurations, des tendances similaires sont observées lors de l’élagage, ce qui atteste d’un schéma de convergence robuste et peu sensible au choix initial.

Les résultats présentés en Figure 9.2 confortent l’hypothèse précédemment établie selon laquelle, à couverture spectrale fixée (largeur de bande non modulable), les solutions obtenues demeurent proches de l’optimum. Ainsi, recourir à des filtres à largeur de bande variable n’entraînerait pas, pour cette tâche, un gain suffisant pour compenser les coûts énergétiques et les contraintes matérielles associées.

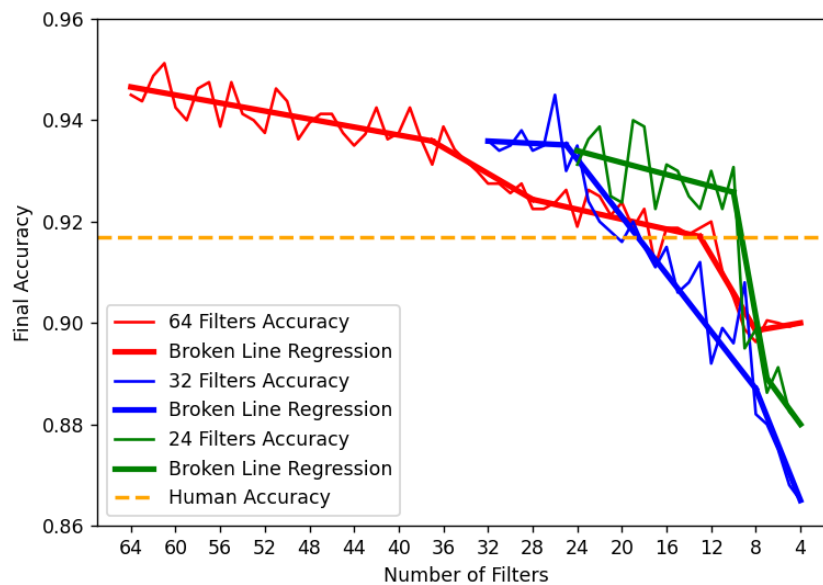


FIGURE 9.2 – Précision en fonction du nombre de filtres, comparée à la performance humaine moyenne (orange). En rouge : tendance avec 64 filtres sur une échelle Mel ; en bleu : tendance avec 32 filtres sur une échelle Mel ; en vert : tendance avec 24 filtres sur une échelle Bark.

9.2 Cas d’application 2 : Reconnaissance d’oiseaux

9.2.1 Comparatif des performances entre différents modèles

Le Tableau 9.3 présente une comparaison des performances moyennes de plusieurs modèles de référence ; en particulier, *BirdNET* et *Perch* (Google) comptent parmi les rares modèles spécifiquement dédiés à la reconnaissance d’oiseaux. À la différence des autres approches, le modèle *EcoPhonic* peut être embarqué directement sur microcontrôleur.

La reconnaissance d’espèces d’oiseaux est une tâche plus complexe que l’application précédente. Cette fois-ci nous comparons des acquisitions réalisées à 100 Hz sur 3 secondes avec des filtres d’ordre 2, 4 et 8. Les autres modèles opèrent quant à eux sur des spectres obtenus numériquement. Les *Performance–Power Index* (PPI) reportés intègrent à la fois le coût énergétique moyen (acquisition + inférence du réseau) et la performance. **Dans cette évaluation, l’impact du bloc de déclenchement n’est pas pris en compte : l’acquisition est supposée permanente.**

TABLE 9.3 – Performance des différents modèles d’IA pour la tâche de reconnaissance d’oiseaux. La taille du spectrogramme d’entrée varie selon le modèle (64×320 pour Yolo). Deux configurations d’*EcoPhonic* sont présentées : version non optimisée (64 filtres) et version élaguée (12 filtres) ($\times 300$). Les MMACs correspondent au nombre d’opérations multiplication-accumulation par inférence ($\times 10^6$). Tous les résultats sont reportés sans quantification.

¹ EcoPhonic-N-2/4/8o emploie un bloc PEx avec des filtres passe-bande d’ordre 2, 4 et 8, respectivement.

² La mémoire d’activations indiquée correspond au pic de consommation RAM obtenu avec les optimisations du compilateur sur *STM32F413ZH*.

Modèle ¹	Fenêtre	MMACs	Memory Usage (MB)		F1-score	PR-AUC	PPI
			Flash	RAM ²			
SSL (AudioMAE) [18]	10 s	2550.0	341.0	NA	80.1 ± 1.4	85.0 ± 0.9	4.69
YOLO_v8m [21]	3 s	2000.0	98.7	2.58	81.3 ± 1.5	86.9 ± 1.3	4.68
Benbogart [17]	5 s	1227.0	72.0	3.39	80.8 ± 1.3	86.1 ± 1.1	4.89
Perch [15]	5 s	820.0	55.1	7.81	90.6 ± 1.1	93.1 ± 0.6	6.80
BirdNET v2.4 [10]	3 s	413.0	50.5	4.72	87.1 ± 1.9	90.4 ± 1.1	6.19
YamNet-1024 [16]	0.96 s	69.2	12.2	0.40	62.0 ± 0.8	68.4 ± 1.0	3.68
YamNet-256 [16]	0.96 s	25.3	0.53	0.40	67.9 ± 0.7	72.9 ± 0.9	4.35
EcoPhonic-64-2o	3 s	16.1	0.27	0.16	69.5 ± 1.9	75.0 ± 2.3	5.18
EcoPhonic-64-4o (nous)	3 s	16.1	0.27	0.16	73.3 ± 1.8	80.6 ± 2.0	5.96
EcoPhonic-64-8o	3 s	16.1	0.27	0.16	77.2 ± 2.1	84.1 ± 2.5	6.46
EcoPhonic-12-2o	3 s	4.5	0.10	0.03	67.7 ± 2.3	73.1 ± 2.6	5.39
EcoPhonic-12-4o (nous)	3 s	4.5	0.10	0.03	72.8 ± 1.8	79.9 ± 1.9	6.41
EcoPhonic-12-8o	3 s	4.5	0.10	0.03	73.3 ± 0.6	80.7 ± 1.1	6.50

9.2.2 Evolution de l’élagage

À l’instar de la tâche KWS, le seuil de 12 filtres constitue, encore une fois, notre point critique. De même, les filtres d’ordre 2 n’apportent pas le meilleur compromis énergie-performance ; on préférera ici l’ordre 4 ou 8.

En pratique, on observe que plus le nombre de filtres est réduit, plus les performances associées à différents ordres de filtrage tendent à se rapprocher. La principale différence réside alors dans le nombre d’époques nécessaires à l’apprentissage pour atteindre ce score. Les bandes retenues restent globalement similaires, mais la stabilité du processus d’élagage se dégrade lorsque l’ordre diminue. Plus précisément, seul l’élagage des filtres d’ordre 8 impacte réellement les performances, avec une dégradation notable entre 24 et 12 filtres (Fig. 9.4).

Afin de réduire l’impact aléatoire dans nos résultats, la même expérience a été conduite plusieurs fois. La Figure 9.3 fournit un indicateur moyen sur l’ordre de suppression : plus l’indice est élevé, plus la bande associée (Tableau 9.4) est éliminée rapidement.

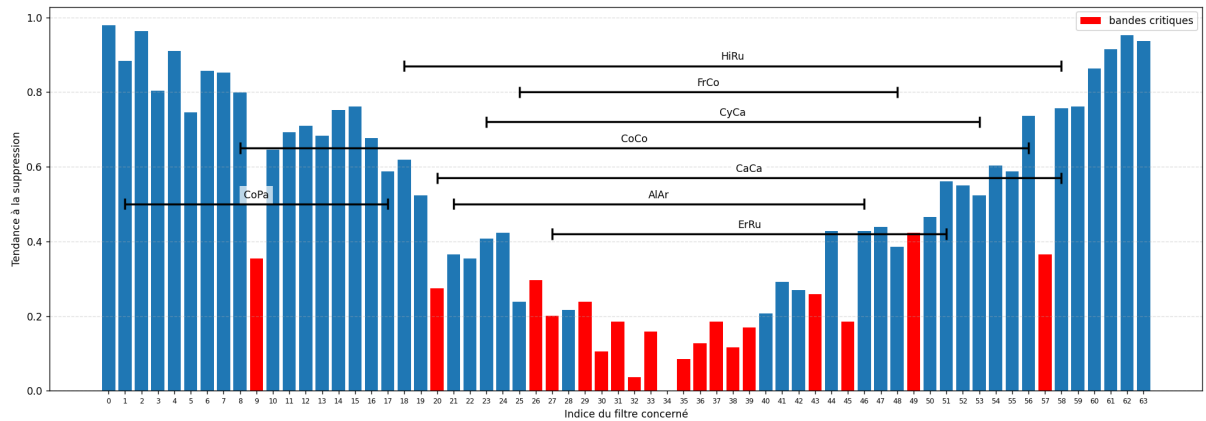


FIGURE 9.3 – Tendance moyennes des filtres à la suppression. En rouge, les bandes fréquentielles ayant terminé parmi les 10 dernières dans plus de 30% des cas.

TABLE 9.4 – Fréquence de coupure des 64 bandes de filtres initiales (échelle Mel) pour la tâche de reconnaissance d’oiseaux. En gras*, les bandes critiques trouvées.

Numéro	Bande Fréq. (Hz)	Numéro	Bande Fréq. (Hz)	Numéro	Bande Fréq. (Hz)
1	50 – 88	23	1566 – 1682	45*	6146 – 6495
2	88 – 129	24	1682 – 1805	46	6495 – 6868
3	129 – 172	25	1805 – 1935	47	6868 – 7261
4	172 – 217	26*	1935 – 2069	48	7261 – 7667
5	217 – 264	27*	2069 – 2213	49*	7667 – 8101
6	264 – 314	28	2213 – 2364	50	8101 – 8557
7	314 – 366	29*	2364 – 2520	51	8557 – 9029
8	366 – 421	30*	2520 – 2687	52	9029 – 9534
9*	421 – 478	31*	2687 – 2860	53	9534 – 10055
10	478 – 540	32*	2860 – 3045	54	10055 – 10613
11	540 – 604	33*	3045 – 3239	55	10613 – 11200
12	604 – 670	34*	3239 – 3440	56	11200 – 11806
13	670 – 741	35*	3440 – 3654	57*	11806 – 12455
14	741 – 816	36*	3654 – 3880	58	12455 – 13137
15	816 – 893	37*	3880 – 4114	59	13137 – 13842
16	893 – 976	38*	4114 – 4363	60	13842 – 14596
17	976 – 1063	39*	4363 – 4626	61	14596 – 15390
18	1063 – 1153	40	4626 – 4897	62	15390 – 16210
19	1153 – 1249	41	4897 – 5188	63	16210 – 17087
20*	1249 – 1348	42	5188 – 5488	64	17087 – 18000
21	1348 – 1454	43*	5488 – 5809		
22	1454 – 1566	44	5809 – 6146		

9.2.3 Analyse biologique

La première phase d’élagage resserre notre domaine sur 400-12 500 Hz, ce qui est cohérent avec nos connaissances des domaines psychoacoustiques des espèces ciblées (zone indiquée en noir sur la figure 9.3).

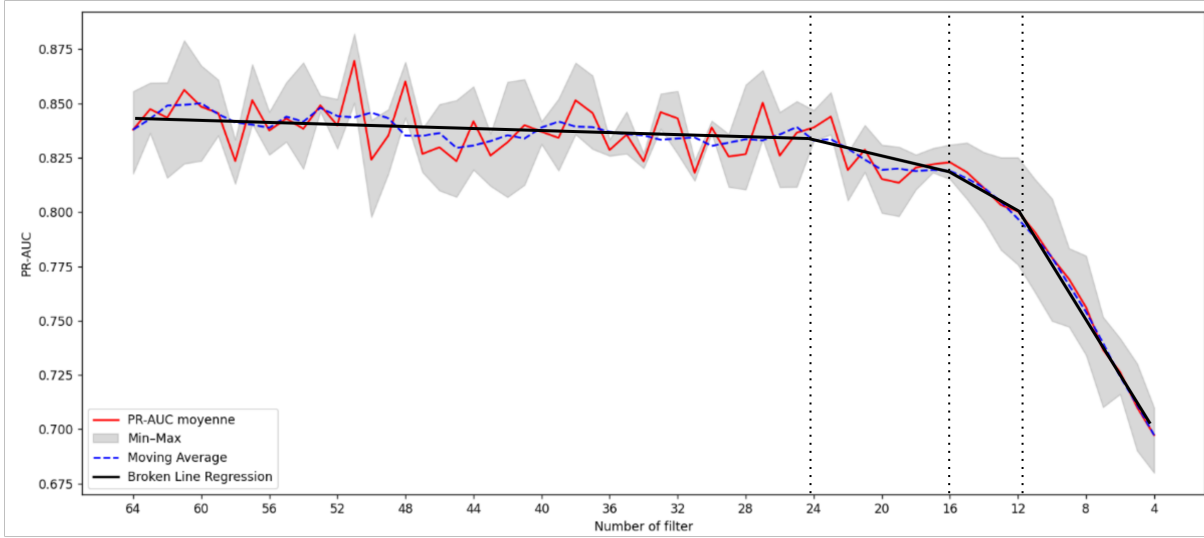


FIGURE 9.4 – PR-AUC moyen du réseau *EcoPhonic* en fonction du nombre de filtres d'ordre 8 (de 64 à 4) pour 10 espèces d'oiseaux.

Dans un second temps, on observe une tendance globale à recentrer la bande fréquentielle autour de 1 900–4 600 Hz, tout en conservant certaines bandes aux caractéristiques particulières. Ce domaine final est pertinent : il est partagé par 9 des 10 espèces ciblées.

Les bandes fréquentielles restantes notables (9, 20, 49 et 57) ont un point commun : elles sont situées aux extrémités de certains des domaines fréquentiels testés. Elles constituent donc des caractéristiques particulièrement utiles pour discriminer les espèces pour le réseau de neurones.

Concrètement, la bande n° 9 (420–480 Hz) se trouve au centre du domaine spectral de l'espèce *CoPa* et lui est propre. La bande n° 57 (autour de 12 000 Hz) n'est atteinte que par *HiRu* et *CaCa*, ce qui les distingue des autres. L'information contenue dans la bande n° 49 exclut les classes *FrCo* et *AlAr*. Même logique en basse fréquence, la bande n° 20 n'est atteinte que par les classes *CaCa*, *CoCo* et *HiRu*.

En conclusion, l'ordre d'élagage mis en évidence par l'algorithme colle avec une intuition experte, laquelle suppose une connaissance poussée du domaine, mais peut paraître bien moins évidente à la première inspection du jeu de test.

9.3 Cas d'application 3 : Présence de chauves-souris

La détection de la présence de chauves-souris est relativement simple : les domaines acoustiques utilisés pour l'écholocalisation ne se superposent à aucun autre signal d'origine biologique.

Dans notre simulation, nous considérons une plage spectrale de 50 Hz à 80 kHz. Les échantillons d'entrée sont extraits à 300 Hz et durant 0.2 s, fournissant ainsi un ensemble d'images de résolution 64×60 .

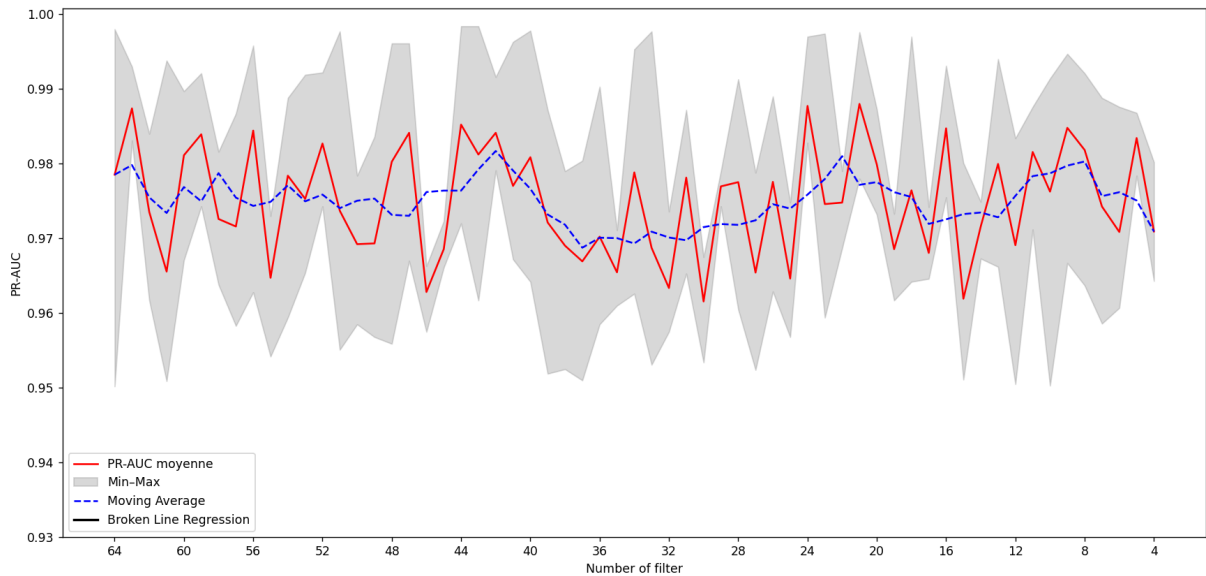


FIGURE 9.5 – PR-AUC moyen du réseau *EcoPhonic* en fonction du nombre de filtres (de 64 à 4) dans le cadre d’une détection de présence de chauves-souris (sortie binaire).

Au cours de l’entraînement du système, on observe un élagage progressif des filtres de plus basse fréquence. Le processus débute par l’atténuation des filtres dans la bande 50–10 000 Hz, connue pour concentrer une forte densité de bruit environnemental et non pertinent, une information trompeuse donc qui est ciblée en priorité. Après cette première réduction du bruit, l’élagage se poursuit de manière plus aléatoire au sein de des plages fréquentielles non pertinentes : 10–35 kHz et 65–80 kHz.

Cette élimination sélective continue jusqu’à ce qu’environ 25 filtres demeurent actifs dans la bande 35–65 kHz, qui correspond à la zone ultrasonore typique des espèces de chauves-souris étudiées. À partir de ce point, les filtres sont supprimés progressivement et de façon relativement uniforme au sein de cette bande résiduelle.

Fait notable, cette phase finale d’élagage n’a pas de réel impact sur les performances : le système maintient une PR-AUC stable d’environ 97.5 %. Étant donné la faible complexité de la tâche, l’objectif n’était pas de maximiser la précision de classification, mais d’observer la manière dont le système converge dans le domaine des hautes fréquences.

Résultat de l'apprentissage du Trigger

10.1 Pertinence des résultats

Les jeux de données utilisés pour la détection de mots-clés et la présence de chauves-souris ne sont pas pertinents pour cette évaluation. Les enregistrements audio sont en moyenne trop courts et l'environnement acoustique trop simple pour constituer un véritable défi pour cette tâche.

Pour appuyer cette observation, nous avons appliqué la première méthodologie décrite en section 7.2.2 dans le cadre de l'expérience de KWS. À l'issue de l'apprentissage, le taux de vrais positifs (TPR) atteignait 99.1 %, tandis que le taux de faux positifs (FPR) demeurait marginal, à seulement 3.0 %.

Ces performances quasi parfaites confirment que la difficulté de la tâche se manifeste surtout en conditions non contrôlées, lorsque les bandes fréquentielles d'intérêt sont partagées avec d'autres sources sonores. En conséquence, l'évaluation de l'apprentissage du bloc *Trigger* est menée exclusivement sur les chants d'oiseaux, contexte dans lequel ces contraintes sont les plus marquées.

10.2 Performance des différentes méthodes envisagées

Comme présenté dans la section 7.2.2, plusieurs stratégies ont été envisagées, toutes reposant sur une inférence à 8 Hz d'un réseau de neurones. Le tableau 10.1 compare les performances obtenues ainsi que le coût énergétique (inférence du réseau) sur notre jeu de données d'oiseaux avec un SNR de 0 dB. Dans ces expérimentations, nous utilisons 12 primitives avec des filtres d'ordre 4, ainsi qu'une primitive dédiée à l'extraction du niveau de bruit courant.

Méthode	Coût (μW)	Précision	Rappel	F1-score
tenseur simple	3.83	0.725	0.779	0.751
lc-PCEN	2.75	0.632	0.750	0.686
tenseur simple (×8)	11.61	0.809	0.813	0.812
lc-PCEN (×8)	9.08	0.659	0.752	0.702

TABLE 10.1 – Comparaison de l’efficacité des différentes méthodes envisagées.

À la lumière de ces résultats, nous retenons la troisième approche : le traitement des huit derniers échantillons contenus dans les 26 *buffers* locaux d’extraction des primitives (flux d’origine et version filtrée en passe-bas) au moyen d’un petit réseau comprenant un GRU et des couches denses.

Nous remarquons par ailleurs que notre tentative de normaliser les flux de primitives par leur version filtrée en passe-bas, une stratégie qui se voulait semblable au PCEN (formule ci-dessous), s’est révélée contre-productive dans ce cadre.

$$\text{PCEN}(x) = \left(\frac{x}{(\varepsilon + M)^\alpha} + \delta \right)^r - \delta,$$

où M correspond à une moyenne glissante du signal d’entrée x . Cette simplification trop rudimentaire a probablement réduit la capacité du modèle à percevoir correctement l’intensité instantanée du signal, pourtant essentielle à l’identification des indices de présence sonore.

Il est important de souligner que la performance du bloc *Trigger* a un double impact. Une précision trop faible provoque un excès de déclenchements du réseau principal, ce qui accroît la consommation énergétique. À l’inverse, un rappel insuffisant conduit à ignorer des événements pertinents qui ne seront alors pas analysés par le CRNN, dégradant ainsi la performance globale du système.

On peut dès lors supposer qu’un ordre de filtre plus élevé ou un nombre accru de primitives puisse améliorer les résultats, même si ces configurations ne correspondent pas au compromis optimal identifié précédemment pour le réseau principal. Afin de vérifier cette hypothèse, le tableau 10.2 présente les performances obtenues avec des filtres d’ordre 4 et 8, ainsi qu’avec 12 et 64 primitives.

Ordre	Nb. de filtres	Coût (μW)	Précision	Rappel	F1-score
8	64	65.08	0.802	0.831	0.816
8	12	11.61	0.810	0.815	0.813
4	64	65.08	0.799	0.827	0.813
4	12	11.61	0.809	0.813	0.812

TABLE 10.2 – Comparaison de l’efficacité des différentes méthodes envisagées.

Les résultats obtenus sont sans équivoque : dans notre cas, ni l’ordre des filtres ni le nombre de primitives n’ont d’impacts significatifs sur les performances. Comme anticipé, l’optimum déterminé pour le CRNN s’avère également pertinent pour le bloc de déclenchement. Il est donc cohérent de conserver cette méthodologie d’optimisation.

Efficacité globale des PrNNs

11.1 Comparatif global

L'utilisation des *Primitive Neural Networks* (PrNN) présente peu d'intérêt lorsque le taux de présence des événements ciblés est élevé. Dans ce cas, le coût énergétique de l'inférence du réseau de neurones devient prépondérant face au coût de l'acquisition analogique. À l'inverse, pour des événements rares, le gain est notable puisque le système reste la plupart du temps en ultra-basse consommation.

Le Tableau 11.1 propose une synthèse des différentes approches¹ pour la tâche de détection d'oiseaux. Les colonnes *consommation du réseau* et *consommation d'acquisition* sont définies différemment selon qu'il s'agit de notre approche (*Ecophonic*) ou de celles de l'état de l'art.

Dans le cas général, la « consommation du réseau » correspond uniquement au coût énergétique de l'inférence du modèle de détection, calculé sur la base d'une inférence par fenêtre d'entrée, sans recouvrement². Pour notre approche, ce coût inclut également l'acquisition à 100 Hz via l'ADC, car il constitue la partie variable de la consommation totale, dépendante du taux de déclenchement du bloc *Trigger*, lui-même fonction du taux de présence. Afin de neutraliser les biais liés au support matériel, toutes les estimations reposent sur l'efficacité énergétique du microcontrôleur *STM32F413ZH*. On peut noter qu'en pratique, ce choix avantage fortement les modèles volumineux, qui s'exécutent généralement sur des plateformes de type Raspberry Pi, beaucoup moins optimisées et dont la consommation atteint typiquement l'ordre du Watt.

Concernant la « consommation d'acquisition », elle désigne, pour les méthodes classiques, l'énergie nécessaire à l'échantillonnage et aux prétraitements. Dans notre approche, elle correspond à la part fixe de la consommation globale : l'extraction des primitives et l'exécution du *Trigger* à 8 Hz en mode *always-on*.

1. La description détaillée des réseaux est donnée en section 1.2.2.

2. La fréquence d'inférence et la résolution temporelle varient donc d'un modèle à l'autre.

Lorsque le taux de présence est de 100%, le *Trigger* n'apporte aucun bénéfice. Il n'est donc pas pris en compte dans notre estimation finale de la consommation et de la performance, ce qui donne un fonctionnement similaire à celui des autres IAs.

Dans les autres scénarii, la mesure de la PR-AUC est réalisée avec la véritable architecture des PrNNs : on prend en compte le bloc de déclenchement dans la chaîne de détection et le CRNN a été réentraîné en conséquence. La performance globale obtenue dépasse le simple produit des performances individuelles du *Trigger* et du CRNN. Cela s'explique, d'une part, par le recouvrement d'une partie des échantillons non détectés, et, d'autre part, par l'alignement des vocalises à gauche de l'image, qui améliore la robustesse de la détection.

La consommation totale est estimée selon la relation suivante :

$$\text{Conso. totale} = [(1 - \text{Précision})(1 - \tau) + \text{Rappel} \times \tau] \times (\text{Conso. réseau}) + \text{Conso. acquisition},$$

où τ désigne le taux de présence (entre 0 et 1 exclu).

On observe que lorsque τ est faible, son influence sur la consommation globale devient marginale : les déclenchements sont alors dominés par les faux positifs, proportionnels à $1 - \text{Précision}$. Ce comportement induit un palier plancher de consommation, en deçà duquel il n'est pas possible de descendre, même en réduisant la fréquence réelle des événements détectés.

Modèle	Conso. du réseau (mW)	Conso. à l'acquisition (mW)	PR-AUC (%)	Taux de présence : 100 %		Taux de présence : 10 %		Taux de présence : 1 %	
				Conso. totale (mW)	PPI	Conso. totale (mW)	PPI	Conso. totale (mW)	PPI
YOLO_v8m	824.70	13.41	86.9	838.11			4.68		
SSL (AudioMAE)	356.60	14.20	85.0	370.80			4.69		
Benbogat	351.00	34.73	86.1	385.73			4.89		
Perch	248.70	26.31	93.1	275.01			6.80		
BirdNET v2.4	214.20	78.17	90.4	292.37			6.19		
YamNet-1024	65.80	12.65	68.4	78.45			3.68		
YamNet-256	27.70	12.65	72.9	40.35			4.35		
EcoPhonic-64-8o	7.50	0.72 + 0.07	84.1 81.0	8.22	6.46	2.75	6.54	2.32	6.63
EcoPhonic-64-4o	7.50	0.58 + 0.07	80.6 77.7	8.08	5.94	2.63	6.18	2.20	6.27
EcoPhonic-12-8o	2.60	0.14 + 0.01	80.7 77.8	2.74	6.50	0.80	6.80	0.66	6.89
EcoPhonic-12-4o	2.60	0.11 + 0.01	79.9 76.8	2.71	6.41	0.78	6.70	0.63	6.81

TABLE 11.1 – Évaluation comparative des modèles d'IA pour la reconnaissance des oiseaux.

11.2 Conclusion de l'analyse

Les contre-performances des modèles *AudioMAE* et *YAMNET* peuvent s'expliquer par le fait que ces modèles sont principalement dédiés à de la reconnaissance humaine et non d'oiseaux, les caractéristiques extraites du signal audio ne sont donc pas fine-tunées pour cette tâche. En particulier, on observe que *YAMNET-1024* a une performance moindre que *YAMNET-256*, ce qui montre que l'augmentation du nombre d'opérations ne garantit pas, à elle seule, une amélioration. Une conclusion assez similaire peut-être faite au sujet de *Yolo*, qui est structurellement plus adapté pour des traitements de grandes images et de détection d'objects.

A partir des démonstrations précédentes, il paraît clair que l'utilisation des PrNNs pour des tâches relativement simples, telles que le *Keyword Spotting*, est bien plus avantageuse que les alternatives présentes dans l'état de l'art. Pour cette tâche, l'usage de 12 filtres d'ordre 4 apparaît optimal et permet d'atteindre le domaine de l'ultra-basse consommation sans recourir à des implémentations ASIC.

A contrario, lorsque l'on considère des jeux de données plus complexes et plus représentatifs de la réalité biologique, tels que la détection acoustique d'oiseaux, la nuance est plus marquée : l'environnement d'acquisition a un impact nettement plus important sur la performance de notre bloc *Trigger* et, par conséquent, sur la performance globale du système par rapport aux alternatives « always-on ». Nous sommes conscient qu'une telle tâche n'est pas parfaitement optimale pour tirer avantage des PrNNs, raison pour laquelle il s'agit d'un exemple parfait pour fixer de nouvelles limites à l'état de l'art.

Ici, la performance du bloc *Trigger* oblige un certain nombre de déclenchements inutiles, cela fixe un nouveau plancher de consommation minimale, plusieurs fois supérieur à celui de l'acquisition analogique et quelque soit le taux de présence effectif. En particulier, nos comparaisons montrent que l'ordre 4 n'offre pas le meilleur compromis énergie-performance. L'augmentation de l'énergie requise par l'acquisition avec un ordre plus élevé n'est absolument pas significative au regard du reste de la consommation et peut valoir le coup même si cela ne fait gagner qu' 1% de performance.

Malgré cette réalité, notre architecture a dépassé le plafond de verre de la consommation énergétique des IAs : elle obtient un compromis performance-énergie similaire au meilleur modèle de l'état de l'art (*Perch*) mais dans un domaine où l'énergie nécessaire est 400 fois inférieure, en dessous du mW.

Ces résultats nous orientent également sur les axes principaux de recherche à venir : cibler la consommation du réseau de neurones principal et la performance du block *Trigger*. Ces perspectives d'amélioration sont détaillées dans le chapitre suivant.

Perspectives

12.1 Sensibilité variationnelle par attention dynamique

Au cours de cette étude, nous n'avons pas pu développer pleinement l'apprentissage du module de sensibilité variationnelle, raison pour laquelle ce dernier est introduit dans la partie dédiée à la construction des PrNNs, mais pas dans la partie des résultats associés. Ce module est le dernier pilier des PrNNs ; sa résolution permettrait de boucler rigoureusement notre analogie avec le fonctionnement biologique. Son intérêt principal est d'optimiser dynamiquement les performances du bloc de déclenchement, limitant ainsi les pertes d'énergies inutiles, en réponse à la limitation observée précédemment. De plus, nous espérons que ce dernier permettra, à l'instar de la biologie, d'améliorer le SNR perçu et, par conséquent, les performances globales.

Après analyse du problème, il apparaît évident que ce module est le plus complexe à implémenter et qu'il nécessitera à lui seul une thèse ou un postdoctorat dédié. Son intérêt vient également en dernier car il n'est pas directement impliqué dans la phase de coévolution structurelle. Pour reprendre l'analogie biologique, cette phase d'apprentissage représente notre auto-évolution, en tant qu'individu, dans une représentation donnée de l'environnement. Elle diffère de l'évolution mimée par les autres modules, qui modélisent notre coévolution, en tant qu'espèce, au sein d'un environnement au sens large.

La principale raison de cette difficulté d'implémentation tient à l'absence, au moment de la rédaction de ce manuscrit, d'un état de l'art établi sur le sujet.

Pour entrer plus en détail, on cherche à moduler dynamiquement l'entrée perçue par notre réseau de neurones, afin d'obtenir une sortie distincte de celle normalement obtenue, plus proche de la réalité du terrain. Cette modulation doit se faire (i) en prenant en compte les reconnaissances précédemment effectuées à moyen terme (contexte), mais aussi, et c'est le plus compliqué, (ii) lors de l'enregistrement courant, c'est-à-dire avant son traitement par le réseau final (hypothèse).

Il s'agit là du mécanisme d'attention biologique¹, un phénomène encore mal compris qui, si l'on extrapole, frôle la notion de conscience.

Au sens strict, cette logique va engendrer une forme d'« hallucination » auditive : ce qui est le but recherché lorsqu'elle nous oriente vers une réponse correcte, mais problématique lorsqu'elle s'éloigne de la réalité². Un tel phénomène impliquera de trouver, avec sa nouvelle méthodologie d'apprentissage³, une métrique qui guidera le modèle vers un juste degré de « soupçon », de manière à ne pas finir avec une IA trop « naïve » qui ne remet pas en question une entrée perturbée, ni « complotiste », qui remettrait en question systématiquement les détections n'étant pas certaines et fabulerait des contre-vérités de manière très catégorique.

Malgré nos tentatives, nous n'avons pas identifié de compromis satisfaisant entre les gains de performance et les coûts computationnels et énergétiques supplémentaires. En l'état, nous retenons donc un simple moteur de règles stochastiques s'appuyant sur une expertise humaine, utilisé comme mécanisme de raisonnement dans l'implémentation.

12.2 Implémentation matérielle

L'intérêt d'une architecture de type PrNN ayant été établi dans ce manuscrit, la prochaine étape consiste à concevoir une carte matérielle mixte analogique-numérique dédiée à son implémentation. À court terme, l'objectif est de valider la preuve de concept au moyen d'un premier prototype pleinement opérationnel, en vue d'un déploiement pilote sur des sites stratégiques et sur de longues périodes.

Ce déploiement en conditions réelles permettra de consolider les performances, d'ajuster l'architecture (robustesse, consommation, coûts) et de préparer la commercialisation.

12.3 Vers une intégration sur silicium

Comme nous l'avons souligné, certaines observations issues de l'état de l'art nous amènent à penser qu'une implémentation ASIC pourrait améliorer le PPI de notre IA en réduisant sa consommation effective d'environ un ordre de grandeur. La cible principale est ici la consommation énergétique du CRNN. On sait déjà, d'après la littérature, que certains microcontrôleurs comme les MAX78000 et GAP9 permettent d'atteindre de tels niveaux d'efficacité. L'objectif du projet ARN-DFG, mené en collaboration avec l'IM2NP, est de quantifier expérimentalement les gains réellement observables grâce à une implémentation matérielle sur mesure [83], et ainsi de déterminer les performances énergétiques maximales atteignables par les PrNNs.

1. À ne pas confondre avec l'« attention » mise en œuvre dans les *Transformers*, qui fonctionne sur une mise en mémoire exhaustive et figée du contexte.

2. Un effet secondaire également observé chez les êtres vivants.

3. Il est nécessaire de s'entraîner sur des échantillons de plusieurs minutes, voir heures, et non des spectrogrammes de quelques secondes comme ce qui a toujours été fait dans l'état de l'art.

Conclusion

Les PrNNs ouvrent la voie à une utilisation de longue durée de systèmes de détection autonomes dans des environnements embarqués. En particulier, cette preuve de concept cible des tâches de reconnaissance acoustique, mais reste transposable à d'autres types de détection sensorielle.

Notre solution constitue une analogie globale du fonctionnement biologique, tout en restant pragmatique et compatible avec les technologies actuelles. L'innovation majeure de cette architecture réside dans sa capacité à reproduire les mécanismes sensoriels biologiques et à simuler un processus de sélection naturelle.

La structure finale d'un PrNN repose sur une captation analogique spécialisée du signal, indépendante de connaissances humaines a posteriori, et bien moins énergivore que les méthodes d'acquisition et de traitements numériques présentes dans l'état de l'art. La partie névralgique du réseau demeure, quant à elle, numérique, mais son utilisation est parcimonieuse tout en restant opérationnelle en permanence. Cette forme hybride tire ainsi parti à la fois de l'efficacité analogique pour les traitements à haute fréquence et de la robustesse des méthodes d'apprentissage numériques pour la classification finale.

Les résultats finaux ont démontré la possibilité de traiter des tâches complexes avec une puissance moyenne inférieure au milliwatt, voire en ultra-basse consommation, tout en atteignant une efficacité globale au moins équivalente à celle des meilleurs modèles de l'état de l'art. Bien que le seuil des 0.25 mW n'ait pas pu être atteint de manière pleinement compétitive pour la tâche de détection d'oiseaux, les estimations issues de ces travaux s'en approchent significativement. Ces recherches ont ainsi permis de dépasser le plafond énergétique imposé par le paradigme actuel et de définir les prochains axes de recherche pour les innovations futures.

En conclusion, les PrNNs répondent pleinement aux objectifs initiaux du projet et amorcent l'émergence d'un nouveau paradigme bio-inspiré pour l'intelligence artificielle embarquée à ultra-basse consommation.

Publications de l’auteur

- [1] V. Gies, S. Marzetti, A. Deverin, B. Valentin, and G. Hervé, “Brevet - COCHLEIA,” Jul 2024.
- [2] A. Deverin, V. Gies, S. Marzetti, and H. Glotin, “Primitive neural networks (PrNN) : A bio-inspired mechanism for reducing power consumption in embedded AI audio processing,” IEEE-ICASSP, May 2026.
- [3] A. Deverin, V. Gies, S. Marzetti, and H. Glotin, “Primitive neural networks (PrNN) : A bio-mimicry paradigm for ultra-low-power AI.” Manuscript submitted for publication to IEEE Access, 2026.
- [4] A. Deverin, V. Gies, S. Marzetti, V. Barchatz, and H. Glotin, “Reducing IA complexity using ear bio-inspired primitives,” *EUSIPCO*, 2024.
- [5] A. Deverin, V. Gies, S. Marzetti, and H. Glotin, “Line gain for long-term environmental survey,” in *One Ocean Science Congress 2025 (OOS)*, (Nice, France). OOS2025-1568.
- [6] A. Deverin, V. Gies, and H. Glotin, “Efficient data augmentation for bioacoustics : A noise-enriched bird sound dataset,” in *IEEE-IOT AIS*.

Bibliographie

- [7] S. M. Warren and P. Walter, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, 1943.
- [8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, 1986.
- [9] W. Koenig, H. K. Dunn, and L. Y. Lacy, “The sound spectrograph,” *The Journal of the Acoustical Society of America*, 1946.
- [10] S. Kahl, C. M. Wood, M. Eibl, and H. Klinck, “BirdNET : A deep learning solution for avian diversity monitoring,” *Ecological Informatics*, vol. 61, p. 101236, 2021.
- [11] C. M. Wood, S. Kahl, A. Rahaman, and H. Klinck, “The machine learning-powered BirdNET App reduces barriers to global bird research by enabling citizen science participation,” *PLOS Biology*, 2022.
- [12] C. Pérez Granados, “BirdNET : applications, performance, pitfalls and future opportunities,” *Ibis*, 02 2023.
- [13] S. Allen-Ankins, S. Hoefer, J. Bartholomew, S. Brodie, and L. Schwarzkopf, “The use of BirdNET embeddings as a fast solution to find novel sound classes in audio recordings,” *Ecology & Evolution*, 2025.
- [14] Xeno-canto Foundation, “Xeno-canto : Sharing bird sounds from around the world.”
- [15] B. Williams, B. van Merriënboer, V. Dumoulin, J. Hamer, E. Triantafillou, A. B. Fleishman, M. McKown, J. E. Munger, A. N. Rice, A. Lillis, C. E. White, C. A. D. Hobbs, T. B. Razak, K. E. Jones, and T. Denton, “Leveraging tropical reef, bird and unrelated sounds for superior transfer learning in marine bioacoustics,” *arXiv*, 2024.
- [16] S. Lachenani, H. Kheddar, and M. Ouldzmirli, “Improving pretrained yamnet for enhanced speech command detection via transfer learning,” in *2024 International Conference on Telecommunications and Intelligent Systems (ICTIS)*, IEEE, 2024.
- [17] B. Bogart, “Bird vocalization classification.” https://github.com/benbogart/bird_vocalization_classification, 2023. Accessed : 2025-06-24.
- [18] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metze, and C. Feichtenhofer, “Masked autoencoders that listen,” *NeurIPS*, 2023.
- [19] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavy, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul 2023.

-
- [20] G. Jocher and U. Team, “YOLOv5 : Pytorch implementation of YOLO (you only look once).” <https://github.com/ultralytics/yolov5>, 2020. Accessed : 2024-11-22.
- [21] D. Reis, J. Kupec, J. Hong, and A. Daoudi, “Real-time flying object detection with YOLOv8,” *ArXiv*, 2024.
- [22] D. O’Shaughnessy, “Speech communication : Human and machine,” *Internet Archive*, 1987.
- [23] S. Umesh, L. Cohen, and D. Nelson, “Fitting the mel scale,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, 1999.
- [24] M. Ravanelli and Y. Bengio, “Speaker recognition from raw waveform with sincnet,” *IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028, 2018.
- [25] N. Zeghidour, O. Teboul, F. de Chaumont Quitry, and M. Tagliasacchi, “LEAF : A learnable frontend for audio classification,” *CoRR*, 2021.
- [26] R. Riad, J. Karadayi, A.-C. Bachoud-Lévi, and E. Dupoux, “Learning spectro-temporal representations of complex sounds with parameterized neural networks,” *The Journal of the Acoustical Society of America*, 2021.
- [27] M. Anderson, T. H. Kinnunen, and N. Harte, “Learnable frontends that do not learn : Quantifying sensitivity to filterbank initialisation,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.
- [28] C. Mead, “Neuromorphic electronic systems,” *Proceedings of the IEEE*, 1990.
- [29] M. A. Mahowald, *VLSI analogs of neuronal visual processing : a synthesis of form and function*. PhD thesis, 1992.
- [30] P. Lichtsteiner, C. Posch, and T. Delbrück, “A 128×128 120 dB 15 μ s latency asynchronous temporal contrast vision sensor,” *IEEE Journal of Solid-State Circuits*, 2008.
- [31] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, “Event-based vision : A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [32] A. Mehonic and A. J. Kenyon, “Brain-inspired computing needs a master plan,” *Nature*, 2022.
- [33] Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal brain damage,” *Advances in Neural Information Processing Systems*, 1990.
- [34] X. Luo, D. Liu, H. Kong, S. Huai, H. Chen, G. Xiong, and W. Liu, “Efficient deep learning infrastructures for embedded computing systems : A comprehensive survey and future envision,” *ACM Trans. Embed. Comput. Syst.*, 2024.
- [35] S. Sun, Y. Chen, and Y. Wang, “On the depth of deep neural networks : A theoretical view,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.
- [36] X. Yu, T. Serra, S. Ramalingam, and S. Zhe, “The combinatorial brain surgeon : Pruning weights that cancel one another in neural networks,” in *International Conference on Machine Learning*, 2022.
- [37] A. Brutzkus and A. Globerson, “Why do larger models generalize better ? a theoretical perspective via the XOR problem,” in *International Conference on Machine Learning*, 2018.

-
- [38] L. Aitchison, “Why bigger is not always better : on finite and infinite neural networks,” *ArXiv*, 2020.
- [39] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural networks,” in *Advances in Neural Information Processing Systems*, 2015.
- [40] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *ArXiv*, 2015.
- [41] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “Fitnets : Hints for thin deep nets,” *CoRR*, 2014.
- [42] S. Zagoruyko and N. Komodakis, “Paying more attention to attention : Improving the performance of convolutional neural networks via attention transfer,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [43] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, 2006.
- [44] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *International Conference on Learning Representations (ICLR)*, 2014.
- [45] M. Mohaimenuzzaman, C. Bergmeir, and B. Meyer, “Pruning vs XNOR-Net : A comprehensive study of deep learning for audio classification on edge-devices,” *IEEE Access*, 2022.
- [46] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [47] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, “Compressing neural networks with the hashing trick,” in *International Conference on Machine Learning (ICML)*, 2015.
- [48] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting linear structure within convolutional networks for efficient evaluation,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.
- [49] Y. Cai, J. Mao, J. Shen, and W. Yang, “Deep learning with hardware acceleration : A survey,” *Integration*, 2018.
- [50] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, “In-datacenter performance analysis of a tensor processing unit,” in *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2017.
- [51] Y. Fu, L. Song, Y. Yuan, C. Zhou, C. Hao, and D. Chen, “Axiomatic pruning for compressing and accelerating neural networks,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [52] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, “Eyeriss : An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” in *IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers*, 2016.
- [53] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, “Efficient processing of deep neural networks : A tutorial and survey,” *Proceedings of the IEEE*, 2017.
- [54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, 1998.

-
- [55] J. L. Elman, “Finding structure in time,” *Cognitive Science*, 1990.
- [56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, 2017.
- [57] K. Cho, B. van Merriënboer, Çağlar Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Conference on Empirical Methods in Natural Language Processing*, 2014.
- [58] J. Chung, Çağlar Gülçehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *ArXiv*, 2014.
- [59] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *International Conference on Learning Representations*, 2017.
- [60] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.
- [61] Y. Liu, Y. Nan, W. Xu, X. Hu, L. Ye, Z. Qin, and P. Liu, “Alphago moment for model architecture discovery,” *ArXiv*, 2025.
- [62] S. Marzetti, V. Gies, V. Barchasz, H. Barthélemy, and H. Glotin, “Comparing analog and digital processing for ultra low-power embedded artificial intelligence,” in *IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*, 2022.
- [63] S. B. Laughlin, “Energy as a constraint on the coding and processing of sensory information,” *Current Opinion in Neurobiology*, 2001.
- [64] V. Saxena, X. Wu, I. Srivastava, and K. Zhu, “Towards neuromorphic learning machines using emerging memory devices with brain-like energy efficiency,” *Journal of Low Power Electronics and Applications*, 2018.
- [65] B. Hayes and J. Wang, “Brain-inspired computing can help us create faster, more energy-efficient computers,” *NIST*, 2022.
- [66] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, and et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, 2014.
- [67] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, and et al., “Loihi : A neuromorphic manycore processor with on-chip learning,” *IEEE Micro*, 2018.
- [68] K. Roy, A. Jaiswal, and P. Panda, “Towards spike-based machine intelligence with neuromorphic computing,” *Nature*, 2019.
- [69] B. Chakravarthi, A. A. Verma, K. Daniilidis, C. Fermüller, and Y. Yang, “Recent event camera innovations : A survey,” in *Computer Vision – ECCV 2024 Workshops*, Springer, 2024.
- [70] P. Warden, “Speech commands : A dataset for limited-vocabulary speech recognition,” *ArXiv*, 2018.
- [71] Y. Zhang, N. Suda, L. Lai, and V. Chandra, “Hello Edge : Keyword spotting on microcontrollers,” *ArXiv*, 2018.
- [72] M. G. Ulkar and O. E. Okman, “Ultra-low power keyword spotting at the edge,” *CoRR*, 2021.

-
- [73] J. Lei, T. Rahman, R. Shafik, A. Wheeldon, A. Yakovlev, O.-C. Granmo, F. Kaw-sar, and A. Mathur, “Low-power audio keyword spotting using tsetlin machines,” *Journal of Low Power Electronics and Applications*, 2021.
 - [74] G. Cerutti, R. Andri, L. Cavigelli, M. Magno, E. Farella, and L. Benini, “Sound event detection with binary neural networks on tightly power-constrained iot devices,” *CoRR*, vol. abs/2101.04446, 2021.
 - [75] Y. Wei, Z. Gong, S. Yang, K. Ye, and Y. Wen, “EdgeCRNN : an edge-computing oriented model of acoustic feature enhancement for keyword spotting,” *Journal of Ambient Intelligence and Humanized Computing*, mar 2022.
 - [76] H. Qin, X. Ma, Y. Ding, X. Li, Y. Zhang, Y. Tian, Z. Ma, J. Luo, and X. Liu, “BiFSMN : Binary neural network for keyword spotting,” in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-2022*, International Joint Conferences on Artificial Intelligence Organization, July 2022.
 - [77] J. Wang and S. Li, “Keyword spotting system and evaluation of pruning and quan-tization methods on low-power edge microcontrollers,” *ArXiv*, 2022.
 - [78] M. Rusci and T. Tuytelaars, “On-device customization of tiny deep learning models for keyword spotting with few examples,” *IEEE Micro*, Nov. 2023.
 - [79] S. Liu, A. van Schaik, B. A. Minch, and T. Delbrück, “Asynchronous binaural spatial audition sensor with $2 \times 64 \times 4$ channel output,” *IEEE Transactions on Biomedical Circuits and Systems*, 2014.
 - [80] S. Marzetti, V. Gies, V. Barchasz, P. Best, S. Paris, H. Barthelemy, and H. Glotin, “Ultra-low power wake-up for long-term biodiversity monitoring,” in *IEEE Inter-national Conference on Internet of Things and Intelligence System (IoTaIS)*, 01 2021.
 - [81] K. Kim, C. Gao, R. Graça, I. Kiselev, H. Yoo, T. Delbruck, and S. Liu, “A 23-pW keyword spotting IC with ring-oscillator-based time-domain feature extraction,” *IEEE Journal of Solid-State Circuits*, 2022.
 - [82] P. Dudek, T. Richardson, L. Bose, S. J. Carey, J. Chen, C. Greatwood, Y. Liu, and W. Mayol-Cuevas, “Sensor-level computer vision with pixel processor arrays for agile robots,” *Science Robotics*, 2022.
 - [83] S. Ray, X. Sun, N. Tremelling, M. Gordiyenko, and P. Kinget, “How tiny can ana-log filterbank features be made for ultra-low-power on-device keyword spotting?,” *CoRR*, 2023.
 - [84] E. Stomatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, “An event-driven classifier for spiking neural networks fed by dynamic vision sensors,” *Frontiers in Neuroscience*, 2017.
 - [85] J. P. Dominguez-Morales, Q. Liu, R. James, D. Gutierrez-Galan, A. Jimenez-Fernandez, S. Davidson, and S. Furber, “Deep spiking neural network model for time-variant signals classification : A real-time speech recognition approach,” in *Pro-ceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*, 2018.
 - [86] V. Algazi, R. Duda, D. Thompson, and C. Avendano, “The CIPIC HRTF database,” in *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, 2001.

-
- [87] B. Gao, Y. Zhou, Q. Zhang, S. Zhang, P. Yao, Y. Xi, *et al.*, “Memristor-based analogue computing for brain-inspired sound localization with in situ training,” *Nature Communications*, 2022.
 - [88] P. M. Hofman, J. G. Van Riswick, and A. J. Van Opstal, “Relearning sound localization with new ears,” *Nature Neuroscience*, 1998.
 - [89] J. Sheaffer, *From Source to Brain : Modelling Sound Propagation and Localisation in Rooms*. PhD thesis, 05 2013.
 - [90] E. A. G. Shaw, *The External Ear*. Springer, 1974.
 - [91] B. J. Marin N, Lobo Cerna F, “Signatures of cochlear processing in neuronal coding of auditory information,” *Mol Cell Neurosci.*, 2022.
 - [92] P. Gilbert, “Chapter 20 - the ear,” in *Physics in the Arts (Third Edition)*, Academic Press, 2022.
 - [93] J. C. Saunders, R. K. Duncan, D. E. Doan, and Y. L. Werner, “The middle ear of reptiles and birds,” in *Comparative Hearing : Birds and Reptiles* (R. J. Dooling, R. R. Fay, and A. N. Popper, eds.), Springer, 2000.
 - [94] A. S. Tucker, “Major evolutionary transitions and innovations : the tympanic middle ear,” *Philosophical Transactions of the Royal Society B : Biological Sciences*, 2017.
 - [95] T. C. Hain and J. Helminski, *Anatomy and Physiology of the Normal Vestibular System*. 2003. Open source book.
 - [96] P. van den Broek, C. Cremers, G. Hoogland, and J. J. Manni, “Anatomy of the auditory pathways, with emphasis on the brain stem,” *Advances in Oto-Rhino-Laryngology*, 1984.
 - [97] Y. Tamura, T. Wada, and H. Liu, “Generating visual information for motion sickness reduction using a computational model based on svc theory,” in *IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 2023.
 - [98] T. Wada, J. Kawano, Y. Okafuji, A. Takamatsu, and M. Mitsuhiro, “A computational model of motion sickness considering visual and vestibular information,” in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020.
 - [99] B. C. J. Moore, “Frequency difference limens for short-duration tones,” *The Journal of the Acoustical Society of America*, 08 2005.
 - [100] H. Fletcher, “The mechanism of hearing as revealed through experiment on the masking effect of thermal noise.,” *Proceedings of the National Academy of Sciences of the United States of America*, 1938.
 - [101] B. Moore, “An introduction to the psychology of hearing,” *Elsevier*, 2004.
 - [102] C. A. Shera, “Laser amplification with a twist : Traveling-wave propagation and gain functions from throughout the cochlea,” *The Journal of the Acoustical Society of America*, 2007.
 - [103] S. S. Narayan, A. N. Temchin, A. Recio, and M. A. Ruggero, “Frequency tuning of basilar membrane and auditory nerve fibers in the same cochleae,” *Science*, 1998.
 - [104] M. A. Ruggero, N. C. Rich, A. Recio, S. S. Narayan, and L. Robles, “Basilar-membrane responses to tones at the base of the chinchilla cochlea,” *The Journal of the Acoustical Society of America*, Apr. 1997.
 - [105] M. Liberman, “Efferent synapses in the inner hair cell area of the cat cochlea : An electron microscopic study of serial sections,” *Hearing Research*, 1980.

-
- [106] H. Spoendlin, “Innervation patterns in the organ of corti of the cat,” *Acta Otolaryngologica*, 1969.
- [107] I. J. Russell and P. M. Sellick, “Low-frequency characteristics of intracellularly recorded receptor potentials in guinea-pig cochlear hair cells,” *The Journal of Physiology*, 1983.
- [108] S. L. Johnson, “Membrane properties specialize mammalian inner hair cells for frequency or intensity encoding,” *eLife*, nov 2015.
- [109] W. E. Brownell, C. R. Bader, D. Bertrand, and Y. de Ribaupierre, “Evoked mechanical responses of isolated cochlear outer hair cells,” *Science*, pp. 194–196, 1985.
- [110] J. Santos-Sacchi and J. Dilger, “Whole cell currents and mechanical responses of isolated outer hair cells,” *Hearing Research*, 1988.
- [111] J. Santos-Sacchi, “On the frequency limit and phase of outer hair cell motility : effects of the membrane filter,” *Journal of Neuroscience*, 1992.
- [112] A. Vavakou, N. P. Cooper, and M. van der Heijden, “The frequency limit of outer hair cell motility measured in vivo,” *eLife*, sep 2019.
- [113] J. B. Dewey, A. Altoè, C. A. Shera, B. E. Applegate, and J. S. Oghalai, “Cochlear outer hair cell electromotility enhances organ of corti motion on a cycle-by-cycle basis at high frequencies in vivo,” *Proceedings of the National Academy of Sciences of the United States of America*, 2021.
- [114] P. Dallos and B. N. Evans, “High-frequency motility of outer hair cells and the cochlear amplifier,” *Science*, 1995.
- [115] S. L. Johnson, M. Beurg, W. Marcotti, and R. Fettiplace, “Prestin-driven cochlear amplification is not limited by the outer hair cell membrane time constant,” *Neuron*, 2011.
- [116] C. Weisz, E. Glowatzki, and P. Fuchs, “The postsynaptic function of type II cochlear afferents,” *Nature*, 2009.
- [117] C. L. Adamson, M. A. Reid, Z. L. Mo, J. Bowne-English, and R. L. Davis, “Firing features and potassium channel content of murine spiral ganglion neurons vary with cochlear location,” *Journal of Comparative Neurology*, Jun 2002.
- [118] Q. Liu and R. L. Davis, “Regional specification of threshold sensitivity and response time in CBA/CaJ mouse spiral ganglion neurons,” *Journal of Neurophysiology*, 2007.
- [119] J. E. Rose, J. F. Brugge, D. J. Anderson, and J. E. Hind, “Phase-locked response to low-frequency tones in single auditory nerve fibers of the squirrel monkey,” *Journal of Neurophysiology*, Jul 1967.
- [120] A. M. Taberner and M. C. Liberman, “Response properties of single auditory nerve fibers in the mouse,” *Journal of Neurophysiology*, Jan 2005.
- [121] J. W. Horst, J. McGee, and E. J. Walsh, “Input–output curves of low and high spontaneous rate auditory nerve fibers are exponential near threshold,” *Hearing Research*, 2018.
- [122] B. Grothe, M. Pecka, and D. McAlpine, “Mechanisms of sound localization in mammals,” *Physiological Reviews*, 2010.
- [123] D. J. Tollin, “The lateral superior olive : a functional role in sound source localization,” *The Neuroscientist*, 2003.
- [124] J. B. Fritz, M. Elhilali, S. V. David, and S. A. Shamma, “Auditory attention—focusing the searchlight on sound,” *Current Opinion in Neurobiology*, 2007.

-
- [125] E. R. Hafter, A. Sarampalis, and P. Loui, “Auditory attention and filters,” in *Auditory Perception of Sound Sources*, Springer, 2008.
 - [126] B. Scharf, J. Magnan, and A. Chays, “On the role of the olivocochlear bundle in hearing : 16 case studies,” *Hearing Research*, 1997.
 - [127] R. R. Ciuman, “The efferent system or olivocochlear function bundle—fine regulator and protector of hearing perception,” *International Journal of Biomedical Science*, 2010.
 - [128] P. Lakatos, G. Musacchia, M. N. O’Connel, A. Y. Falchier, D. C. Javitt, and C. E. Schroeder, “The spectrotemporal filter mechanism of auditory selective attention,” *Neuron*, 2013.
 - [129] Q. Gehmacher, P. Reisinger, T. Hartmann, T. Keintzel, S. Rösch, K. Schwarz, and N. Weisz, “Direct cochlear recordings in humans show a theta rhythmic modulation of auditory nerve activity by selective attention,” *Journal of Neuroscience*, 2022.
 - [130] J. Schnupp, I. Nelken, and A. King, *Auditory Neuroscience : Making Sense of Sound*. MIT Press, 2011.
 - [131] M. Rahman, B. D. B. Willmore, A. J. King, and N. S. Harper, “Simple transformations capture auditory input to cortex,” *Proceedings of the National Academy of Sciences*, 2020.
 - [132] V. Gies, “Toward ultra-low power embedded artificial intelligence.” Habilitation à diriger des recherches (HDR), Université de Toulon, Nov. 2021. Laboratoire IM2NP (UMR 7334).
 - [133] Y. Fu, L. Song, Y. Yuan, C. Zhou, C. Hao, and D. Chen, “SACoD : Sensor–algorithm co-design for phase-layer coded imaging,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
 - [134] Y. Feng, T. Ma, Y. Zhu, and X. Zhang, “BlissCam : Boosting eye tracking efficiency with learned in-sensor sparse sampling,” *ISCA*, 2024.
 - [135] G. Gallego, T. Delbruck, G. Orchard, R. Tóth, *et al.*, “Event-based vision : A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
 - [136] R. Sarpeshkar, *Ultra Low Power Bioelectronics : Fundamentals, Biomedical Applications, and Bio-Inspired Systems*. Cambridge University Press, 2010.
 - [137] F. Chen, K.-F. Un, W.-H. Yu, P.-I. Mak, and R. P. Martins, “A 108-nW 0.8-mm² analog voice activity detector featuring a time-domain CNN with sparsity-aware computation and sparsified quantization in 28-nm CMOS,” *IEEE Journal of Solid-State Circuits*, 2022.
 - [138] H. Fletcher, *Speech and Hearing in Communication*. D. Van Nostrand Company, 1953.
 - [139] J. B. Allen, “Harvey fletcher’s role in the creation of communication acoustics,” *The Journal of the Acoustical Society of America*, 1996.
 - [140] B. J. Marin N, Lobo Cerna F, “Signatures of cochlear processing in neuronal coding of auditory information,” *Mol Cell Neurosci.*, 2022.
 - [141] S. Wang, Z. Li, X. Wang, J. Li, X. Wang, J. Chen, Y. Li, C. Wang, and L. Qin, “Cortical and thalamic modulation of auditory gating in the posterior parietal cortex of awake mice,” *Cerebral Cortex*, 2023.
 - [142] A. G. Samelli and E. Schochat, “The gaps-in-noise test : Gap detection thresholds in normal-hearing young adults,” *International Journal of Audiology*, May 2008.

-
- [143] L. Beker, Y. Chen, and Y. Zhao, "Design of mems energy harvesters for biomedical applications," *Sensors and Actuators A : Physical*, 2013.
- [144] E. Ilik and M. Basdogan, "Piezoelectric micro-actuated cochlear implants for hearing restoration," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2018.
- [145] H. Ulsan, A. Yildiz, and M. Ozkan, "A fully custom ultra-low-power cochlear implant architecture for autonomous operation," *IEEE Transactions on Biomedical Circuits and Systems*, 2024.
- [146] J. Moosmann, H. Müller, N. Zimmerman, G. Rutishauser, L. Benini, and M. Magno, "Flexible and fully quantized lightweight TinyissimoYOLO for ultra-low-power edge systems," *IEEE Access*, 01 2024.
- [147] H. Yang, J.-H. Seol, R. Rothe, Z. Fan, Q. Zhang, H.-S. Kim, D. Blaauw, and D. Sylvester, "A 1.5- μ W fully-integrated keyword spotting SoC in 28-nm CMOS with skip-RNN and fast-settling analog frontend for adaptive frame skipping," *IEEE Journal of Solid-State Circuits*, 01 2023.
- [148] F. Martinelli, G. Dellaferrera, P. Mainar, and M. Cernak, "Spiking neural networks trained with backpropagation for low power neuromorphic implementation of voice activity detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.
- [149] D. H. Horrocks, "Component value selection for active filters using genetic algorithms," in *Proc. IEEE Workshop on Natural Algorithms in Signal Processing*, pp. 13–1, 1993.
- [150] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, 01 1997.
- [151] M. F. Ahmad, N. A. Mat Isa, W. H. Lim, and K. M. Ang, "Differential evolution : A recent review based on state-of-the-art works," *AEJ - Alexandria Engineering Journal*, 05 2022.
- [152] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Parallel Problem Solving from Nature - PPSN VIII*, Springer Berlin Heidelberg, 2004.
- [153] O. Ait Elhara, A. Auger, and N. Hansen, "Permuted Orthogonal Block-Diagonal Transformation Matrices for Large Scale Optimization Benchmarking," in *GECCO 2016*, July 2016.
- [154] K. Varelas, A. Auger, D. Brockhoff, N. Hansen, O. A. Elhara, Y. Semet, R. Kassab, and F. Barbaresco, "A Comparative Study of Large-scale Variants of CMA-ES," in *PPSN XV 2018 - 15th International Conference on Parallel Problem Solving from Nature*, Sept. 2018.
- [155] J.-H. Han, D.-J. Choi, S.-U. Park, and S.-K. Hong, "Hyperparameter optimization for multi-layer data input using genetic algorithm," in *IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*, 2020.
- [156] D. Smilkov, S. Carter, D. Sculley, S. Romantsev, J. Mullins, and J. Wexler, "Direct-manipulation visualization of deep networks." <https://playground.tensorflow.org>, 2017.

-
- [157] K. J. Piczak, “ESC : Dataset for environmental sound classification,” in *Proceedings of the 23rd ACM International Conference on Multimedia*, Association for Computing Machinery, 2015.
- [158] L. R. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice Hall, 1993.
- [159] K. N. Stevens, *Acoustic Phonetics*. MIT Press, 1998.
- [160] E. Zwicker and H. Fastl, *Psychoacoustics : Facts and Models*. Springer, 2006.
- [161] P. Ladefoged, *Vowels and Consonants : An Introduction to the Sounds of Languages*. Blackwell Publishing, 2001.
- [162] M. N. Miah and G. Wang, “Keyword spotting with deep neural network on edge devices,” in *IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2022.
- [163] W. Shan, M. Yang, T. Wang, Y. Lu, H. Cai, L. Zhu, J. Xu, C. Wu, L. Shi, and J. Yang, “A 510-nW wake-up keyword-spotting chip using serial-FFT-based MFCC and binarized depthwise separable CNN in 28-nm CMOS,” *IEEE Journal of Solid-State Circuits*, 2021.
- [164] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, “Efficient keyword spotting using dilated convolutions and gating,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- [165] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, *et al.*, “WaveNet : A generative model for raw audio,” *ISCA*, 2016.
- [166] R. Miiikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. S. Shahrzad, and B. Hodjat, “Evolving deep neural networks,” *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, 2019.
- [167] B. Xue, M. Zhang, W. N. Browne, and X. Yao, “A survey on evolutionary computation approaches to feature selection,” *IEEE Transactions on Evolutionary Computation*, 2015.
- [168] R. A. A. Ince, S. Panzeri, and C. Kayser, “Neural codes formed by small and temporally precise populations in auditory cortex,” *Journal of Neuroscience*, 2013.
- [169] T. Hromádka, M. R. DeWeese, and A. M. Zador, “Sparse representation of sounds in the unanesthetized auditory cortex,” *PLOS Biology*, 2008.
- [170] A. D. Reyes, “Mathematical framework for place coding in the auditory system,” *PLOS Computational Biology*, 2021.