

Titre: Développement d'un système de stimulation en temps réel de l'activité cérébrale
Title: l'activité cérébrale

Auteur: Nicolas Valenchon
Author:

Date: 2021

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Valenchon, N. (2021). Développement d'un système de stimulation en temps réel de l'activité cérébrale [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie.
Citation: <https://publications.polymtl.ca/9138/>

Document en libre accès dans PolyPublie

Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/9138/>
PolyPublie URL:

Directeurs de recherche: Giovanni Beltrame
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Développement d'un système de stimulation en temps réel de l'activité
cérébrale**

NICOLAS VALENCHON

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2021

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Développement d'un système de stimulation en temps réel de l'activité
cérébrale**

présenté par **Nicolas VALENCHON**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Gabriela NICOLESCU, présidente

Giovanni BELTRAME, membre et directeur de recherche

Samuel KADOURY, membre

REMERCIEMENTS

J'aimerais commencer par remercier Giovanni Beltrame, et tous les membres du laboratoire MIST de Polytechnique Montréal, pour leur expertise, leur aide et leur collaboration, en particulier Yann Bouteiller qui a permis d'amener ce projet à son terme, principalement par sa connaissance en apprentissage profond.

Je remercie Hugo Jourde et Emily Coffey d'avoir partagé leur savoir en neuroscience et pour la collaboration que nous avons mené, malgré la distance, depuis le début de ce projet.

Je remercie aussi mon école française, l'Université de Technologie de Compiègne ainsi que Polytechnique Montréal pour avoir rendu possible ce double diplôme.

Finalement, je souhaite remercier ma famille et mes amis de m'avoir soutenu lors de mes études.

RÉSUMÉ

Près d'un tiers de notre vie est consacré au sommeil, et pourtant la connaissance que nous en avons est encore limitée. Que se passe-t-il vraiment pendant cet état inconscient ? L'avancée des technologies du siècle dernier a permis de grandes découvertes dans ce domaine avec le développement de l'électroencéphalographie et les premières analyses de l'activité cérébrale, afin de comprendre au mieux les fonctions cognitives et les états cérébraux. Certaines oscillations, comme les «sleep spindles», jouent un rôle dans le processus de consolidation de la mémoire. Cependant, un lien de causalité reste encore à étudier : la stimulation des «sleep spindles» va-t-elle avoir un impact sur cette consolidation ? Pour répondre à cette question, il faut être capable de stimuler ladite oscillation pendant sa période d'apparition. Cette durée, inférieure à 2,5 secondes, rend la tâche difficile. Un appareil précis et rapide est nécessaire, qui pourra s'adapter à d'autres applications.

De plus, pour permettre des expériences à plus long terme et dans un environnement plus familier pour les sujets, à savoir chez eux, le système doit être portable. Enfin, pour permettre son utilisation par le plus grand nombre, le coût n'est pas à négliger. Il doit aussi être en libre accès pour tous. Aucun appareil ne répond actuellement à ces critères. C'est pour cela que ce projet de maîtrise existe. Pour concevoir le Portiloop : un appareil de stimulation en temps réel de l'activité cérébrale, à faible coût et portable. Trois étapes le composent : l'acquisition du signal EEG, la détection de l'oscillation et sa stimulation. La première et la troisième sont des fonctionnalités nécessaires de l'appareil, mais ne représentent pas une avancée scientifique. La deuxième a été étudiée par de nombreux chercheurs, avec différentes approches. Cependant, le critère temps-réel, ainsi que la volonté de précision et le coût du projet, réduit l'ensemble des solutions applicables. Pour répondre au mieux à toutes ces problématiques, le détecteur est un réseau de neurones artificiel (ANN) implémenté en circuit logique programmable (FPGA).

Le Portiloop permet de réaliser la stimulation des «sleep spindles» en \sim 300 ms après le début de l'oscillation, avec 71% de précision et de rappel sur un jeu de données annotées par des experts. Il pourra donc être utilisé dans de futures recherches sur le rôle et la nature de ces oscillations, ainsi que pour de multiples applications en neurosciences, qui ne se résument pas à l'étude du cas présentée dans ce mémoire.

ABSTRACT

Sleeping represents a third of our life, yet our knowledge of it is still limited. What really happens during this unconscious state? Technological advances in the last century have led to major discoveries in this area with the development of electroencephalography and the first analyses of brain activity, in order to understand better brain state and cognitive functions. Certain waves, like sleep spindles, are known to play a role in the process of memory consolidation. However, a causal link has yet to be studied : will stimulation of the oscillation have an impact on this consolidation? To answer this question, we must be able to stimulate sleep spindles during its onset period. This duration, less than 2.5 seconds, makes the task difficult. A precise and fast device is needed, and should be adaptable to other applications.

Moreover, to allow long-term experiments in an environment more familiar to the subjects, *i.e.*, at home, the system must be portable. Finally, to permit a global use of the device, the low cost is not to be neglected, nor its open access. No previous device meets these criteria. It is the reason why this thesis project exists. To create the Portiloop : a device capable of stimulating in real time the cerebral activity, at low cost and being portable. It is composed of three main functionalities : the acquisition and processing of the EEG signal, the detection of the pattern of interest and its stimulation. The first and third are necessary functions of the device, but do not represent a scientific breakthrough. The second one has been studied by many researchers, with different approaches. However, the real-time criterion, as well as the desire for precision and the low-cost of the project, limit the applicable solutions. To answer all these problems, the detector is an artificial neural network (ANN) implemented in programmable logic circuits (FPGA).

The Portiloop allows to realize the stimulation in ~ 300 ms after the beginning of the oscillation, with 71% of precision and recall on a dataset annotated by experts. It can therefore be used in future research to understand the role and nature of these oscillations, as well as for various other applications in neuroscience, that might not be related to the case study of this thesis.

TABLE DES MATIÈRES

REMERCIEMENTS	iii
RÉSUMÉ	iv
ABSTRACT	v
TABLE DES MATIÈRES	vi
LISTE DES TABLEAUX	ix
LISTE DES FIGURES	x
LISTE DES SIGLES ET ABRÉVIATIONS	xiv
CHAPITRE 1 INTRODUCTION	1
1.1 Définitions et concepts de base	3
1.1.1 Communication sérielle	3
1.1.2 Circuit logique programmable - FPGA	3
1.1.3 Microcontrôleur	4
1.1.4 Convertisseur Analogique-Numérique - ADC	4
1.1.5 Réseau de neurones artificiels	5
1.2 Éléments de la problématique	6
1.2.1 Acquisition d'un signal exploitable	6
1.2.2 Détection en temps réel d'oscillations rapides	6
1.2.3 Système embarqué fonctionnant sur batterie	7
1.2.4 Stimuler l'activité cérébrale	7
1.3 Objectifs de recherche	7
1.4 Plan du mémoire	8
CHAPITRE 2 REVUE DE LITTÉRATURE	9
2.1 Acquisition du signal EEG	9
2.2 Stimulation en boucle fermée	10
2.3 Détection des « sleep spindles »	11
2.4 Réseau de neurones en FPGA	13
2.5 Optimisation multiobjectif	14

CHAPITRE 3 MÉTHODE DE RECHERCHE ET ORGANISATION DU MÉMOIRE	15
3.1 Acquérir un signal exploitable	15
3.2 Détection en temps réel des «sleep spindles»	15
3.3 Structure du Document	16
CHAPITRE 4 ARTICLE 1 : THE PORTILOOP : A DEEP-LEARNING-BASED OPEN-SCIENCE TOOL FOR CLOSED-LOOP BRAIN STIMULATION	17
4.1 Introduction	18
4.2 Related work	19
4.2.1 Devices for EEG acquisition and stimulation	19
4.2.2 Offline sleep spindle detection	20
4.2.3 Online sleep spindle detection	20
4.3 Methods	21
4.3.1 Real-time constraints	21
4.3.2 Architecture of the proposed system	23
4.3.3 Type of ANN	26
4.3.4 Pareto-optimal architecture search	26
4.3.5 Virtual ANN parallelization with time dilation	30
4.3.6 Case study : sleep spindle auditory stimulation	31
4.4 Experiments	34
4.4.1 Online detection	35
4.4.2 Real-time stimulation	40
4.5 Discussion and future work	43
4.6 Conclusion	44
4.7 Appendices	45
4.7.1 Model-based explanation of sleep spindles	45
4.7.2 Stimulation visualizations	47
4.7.3 PMBO hyperparameters	48
4.7.4 Model hyperparameters	49
4.7.5 Best threshold for classifiers and regressors	50
4.7.6 Detection delay distributions	51
4.7.7 2-input network architecture	51
CHAPITRE 5 DISCUSSION GÉNÉRALE	52
CHAPITRE 6 CONCLUSION	55
6.1 Synthèse des Travaux	55

6.2 Limitations	55
6.3 Améliorations Futures	55
RÉFÉRENCES	57

LISTE DES TABLEAUX

4.1	Quantitative results. The nomenclature is “mean (std)” and the super- scripts are for referencing rows in the text.	35
4.2	Hyperparameters used for PMBO	48
4.3	Hyperparameters used to train the final model	49

LISTE DES FIGURES

1.1	Utilisation en situation réelle du Portiloop avec stimulation auditive	2
1.2	Exemple de configuration SPI pour une communication entre un maître et son périphérique.	3
2.1	Signal EEG pendant le sommeil	9
4.1	Main functionalities. Timing is reported from our case study. The constant delay in stimulation output (4 ms for auditory stimuli) enables precise experimental control stimulation timing relative to neural oscillatory phase.	21
4.2	Software accumulation effect of recurrent units. Dark blue signal : no phenomenon of interest. Black signal : phenomenon of interest not detected by the ANN. Green signal ; phenomenon of interest correctly detected. Magenta : output of the ANN. Horizontal grey : detection threshold. The ANN introduces a software delay : the recurrent units of the ANN act as pseudo-accumulators, thus the detection happens with a variable delay.	22
4.3	Detailed architecture of the Portiloop device. (1) The user controls the device from a simple Web User Interface. (2) An EEG front-end device acquires biosignals from the electrodes. (3) The Microblaze soft processor centralizes all operations performed within the FPGA. (4) The Microblaze core sends the digital signal to be filtered through Finite Impulse Response filters. (5) The Microblaze core sends the digital filtered signal to an artificial neural network. The neural network returns the likelihood of the input signal being the desired pattern in order to decide whether a stimulus should be sent.	23
4.4	PMBO. The algorithm is based on a single producer and multiple consumers architecture. The meta learner is in charge of producing relevant hyperparameter sets in a guided fashion. Then, it sends them to idle workers, and keeps producing new sets as long as idle workers remain. Each worker that has received a new set starts training the corresponding ANN. Once this training ends, the real costs of the hyperparameter set can be computed and are sent back to the meta learner. 29	

4.5	Time dilation. In this example, a sliding window of the 4 last samples (underlined with a dotted curve) is used as input to the model. The time dilation is the number of samples between two forward passes in the ANN. When it is small (top), two consecutive windows overlap for the most part (see <i>e.g.</i> , green and red windows). This implies that the recurrent hidden state which persists from one forward pass to the next (arrows) contains a lot of redundancy with the next input. When the time dilation is big (bottom), this issue is corrected, and back-propagation will reach much further back in time for the same number of forward passes. NB : forward passes happen only at the end of arrows in this diagram, the ANN is idle during other time-steps.	30
4.6	Virtual parallelization. In this example, the time dilation is 2. Thus, we keep track of 2 independent hidden states and feed these alternately to the recurrent units of the ANN. This is equivalent to having two decoupled models that are used alternately for inference.	31
4.7	Signal processing pipeline extracting relevant inputs for the ANN. Power features are computed offline only for the SpindleNet architecture and are not represented in this diagram.	33
4.8	Single-input architecture search with PMBO. The hardware cost is the number of trainable parameters in the neural architecture, and the software cost is $1 - f_1$ -score of the fully-trained model. Black : non-Pareto-optimal models tested by the algorithm. Red dots : Pareto-optimal models found by the algorithm. Red line : Pareto front.	37
4.9	Final single-input ANN architecture. The dimensions of each layer are provided in parenthesis using the PyTorch nomenclature.	38
4.10	Stimulation with a 0.5 threshold. The color code is the same as Figure 4.2, with false positives being additionally displayed in red. Note the first part of the spindle is not detected (false negative), which introduces an ANN software delay. In addition, a small portion of the signal after the spindle is still detected as such (false positive), but it has no impact on our stimulation procedure. Finally, another portion of the signal that was not annotated as a spindle by MODA is detected as a spindle by our model (false positive), generating an undesirable stimulus is generated.	40
4.11	Stimulation with a 0.84 threshold.	41

4.12	Evolution of the actual stimulation performance w.r.t. the chosen detection threshold.	41
4.13	Distribution of stimulation delays for a classifier with 0.5 and 0.84 threshold. Delays are negative when spindles are stimulated in advance.	42
4.14	Integrated gradients (classifier ANN output : 0.136). The <i>integrated gradients</i> algorithm enables exploring why the model takes a given decision (the more a portion of the signal is represented in red, the highest its influence on the current output of the ANN). Grey windows are past inputs, whereas the black window is the current input : the past influences the current output due to the RNN. Here, the model finds that it is looking at the aftermath of a sleep spindle. With our time dilation and window size, a small portion of the window overlaps from one sample to the next. We see that this portion (at the left-hand side of each window) is in fact ignored by the model. Therefore, it is probably possible to shrink our model even more, although PMBO did not find this. In the future, this type of visualizations might also help experts better understand what sleep spindles are by revealing unknown influences.	45
4.15	Integrated gradients (classifier ANN output : 0.639). The current window is within an actual sleep spindle. The model mainly focuses on the spindle itself, but also a few events that happened further back in time, to make its decision.	46
4.16	Different stimulation failure modes (classifier with threshold 0.5). Blue : no sleep spindle and no detection. Black : sleep spindle not detected. Green : sleep spindle correctly detected. Red : detection where the signal is not a spindle. Vertical blue : beginning of a spindle. Vertical grey : stimulation. Magenta : ANN output. Horizontal grey : detection threshold. This figure illustrates typical ‘failure’ cases of our final sleep spindle stimulating device. False negative : the first spindle is missed because the threshold is too big. True positive : the second spindle is correctly stimulated. False positive : a part of the signal not labeled as a sleep spindle by MODA is detected as a spindle by the device and stimulated. Almost true positive : the sleep spindle is stimulated in advance (NB : we count this case as a false positive).	47

4.17	Classifier with threshold 0.84, success example. Same color code as Figure 4.16. Increasing the detection threshold from 0.5 to 0.84 removes most false positive stimuli (vertical grey not following vertical blue).	47
4.18	Classifier with threshold 0.84, failure example. Same color code as Figure 4.16. Increasing the detection threshold comes with more false negative stimuli (vertical blue not followed by vertical grey).	48
4.19	F1 score evolution with threshold variation on classification	50
4.20	F1 score evolution with threshold variation on regression	50
4.21	Stimulation histogram for 1 input network classification with a 0.25 threshold	51
4.22	2-input neural network architecture. This architecture consists of two sequences of CNN and RNN. The first sequence processes the cleaned raw signal (input 1), whereas the second processes the envelope (input 2). The latent features from both branches are concatenated and fed to a fully connected layer to yield the output of the ANN.	51
5.1	Face avant du PCB réalisé par © Xavier L'Heureux.	53
5.2	Face arrière du PCB réalisé par © Xavier L'Heureux.	54

LISTE DES SIGLES ET ABRÉVIATIONS

ANN	Artificial Neural Network / Réseau de Neurones Artificiels
ADC	Analog to Digital Converter / Convertisseur Analogique-Numérique
CNN	Convolutional Neural Network / Réseau de Neurones Convolutif
DMA	Direct Memory Access
EEG	Electroencéphalographie
FIR	Finite Impulse Response / Filtre à Réponse Finie
FPGA	Field Programmable Gate Array
RNN	Recurrent Neural Network / Réseau de Neurones Récurent
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
TPU	Tensor Processing Unit
PC	Personal Computer / Ordinateur Personnel
PCB	Printed Circuit Board / Circuit imprimé

CHAPITRE 1 INTRODUCTION

Ce mémoire présente le travail de recherche accompli au sein du laboratoire MIST de Polytechnique Montréal (Montréal, Québec, Canada) entre janvier 2020 et juillet 2021. Ce document est au format d'un mémoire par article, qui a été soumis le 12 août 2021 et est présenté chapitre 4 :

- Nicolas Valenchon, Yann Bouteiller, Hugo R. Jourde, Emily B.J. Coffey, and Giovanni Beltrame, «The Portiloop : a deep learning-based open science tool for closed-loop brain stimulation,» *IEEE Transactions on Biomedical Circuits and Systems*, 2021.

Sous la direction de Giovanni Beltrame, ce travail a été réalisé en collaboration avec le laboratoire CLASP de l'Université de Concordia (Montréal, Québec, Canada). Hugo Jourde, sous la direction d'Emily Coffey, était responsable de la partie neurosciences. Yann Bouteiller a participé activement au développement et à l'entraînement du réseau de neurones.

De plus, il a été financé par le programme Audace du Fond de Recherche du Québec : Santé (Subvention 279561).

L'Institutional Review Board de Concordia a autorisé le 11 janvier 2021, le protocol 30014351, «Validating a real-time sleep spindle detector», qui permet l'utilisation d'un jeu données humains.

L'activité électrique à l'intérieur du cerveau est à la base de la perception, de la pensée et du comportement. Elle peut être capturée sur le cuir chevelu en utilisant l'électroencéphalographie (EEG), à l'aide d'électrodes qui vont mesurer l'activité électrique des neurones. Des études ont été menées depuis le siècle dernier afin de comprendre la relation entre les motifs et les fréquences mesurées dans l'EEG et les fonctions cognitives ainsi que les états cérébraux. Mais il n'est pas possible d'établir un lien de causalité entre ces éléments simplement en les observant. Pour cela, il faut pouvoir interagir avec les oscillations du cerveau à l'aide de stimuli qui peuvent prendre plusieurs formes, dont la stimulation auditive. Grâce à ces études, il est possible de mieux comprendre leur rôle, et de proposer des solutions permettant de restaurer des processus abîmés par l'âge ou la maladie. Cependant pour permettre ces recherches, des outils suffisamment puissants et facilement accessibles sont nécessaires afin de réaliser de la stimulation en boucle fermée. Ce mémoire présente le Portiloop, un appareil de stimulation en boucle fermée de l'activité cérébrale. Il est mis en application avec un cas pratique : la stimulation auditive des «sleep spindles ».

Ces oscillations, ayant une fréquence élevée (\sim 12-16 Hz) par rapport à l'activité principale

du cerveau ($\sim 0.5\text{-}20$ Hz), jouent un rôle dans la consolidation de la mémoire. Cependant, leur durée assez courte ($<2,5$ s) demande une précision importante pour la stimulation, et les méthodes utilisées pour d'autres oscillations ne peuvent pas s'appliquer ici. De plus, même la détection hors-ligne, qui consiste à avoir l'entièreté du signal électrique, passé comme futur, n'arrive pas à mettre d'accord à plus de 70% [1], ni les experts, ni les algorithmes déjà existants.

L'objectif de cette maîtrise est de concevoir et de vérifier un appareil basé sur l'apprentissage profond, portable, fonctionnant sur batterie et à bas coût, qui pourra permettre d'acquérir un signal EEG en temps réel, de détecter des motifs importants pour la recherche en neuroscience et de répondre rapidement avec une stimulation précise. De plus, ce projet permet d'introduire un réseau de neurones artificiels implémenté en FPGA pour détecter des «sleep spindles» et un algorithme de recherche parallélisé pour en optimiser les paramètres. Ce réseau pourra, comme le reste de l'appareil, être adapté à d'autres applications futures. Le Portiloop a pour but d'aider à accélérer la recherche en neuroscience sur la stimulation en boucle fermée en créant ce système et en donnant un accès libre à tous. La Figure 1.1 illustre l'utilisation du prototype de l'appareil dans le cas d'une stimulation auditive.



Figure 1.1: Utilisation en situation réelle du Portiloop avec stimulation auditive

Les contributions de cette maîtrise sont : la réalisation de l'appareil, l'acquisition, le traitement du signal, une participation active à l'algorithme d'exploration, à la conception et à l'entraînement du réseau de neurones, l'implémentation de ce dernier en FPGA et la stimu-

lation auditive.

1.1 Définitions et concepts de base

Avant d'aller plus loin dans ce document, il est important de revenir sur des concepts de base qui vont être utilisés et considérés comme connus par la suite.

L'appareil présenté au chapitre 4, le Portiloop, est un microcontrôleur basé sur une carte Pynq Z2, contenant un FPGA, communiquant à l'aide du protocole SPI avec un convertisseur analogique numérique qui est spécialisé dans l'acquisition de signal EEG. À l'intérieur du FPGA, un réseau de neurones artificiels est implémenté.

1.1.1 Communication sérielle

Il existe plusieurs protocoles de communication sérielle. Dans ce projet seul le Serial Peripheral Interface (SPI) est utilisé. Ce bus de données synchrone permet une communication full-duplex entre un maître et un périphérique, comme le représente la Figure 1.2.

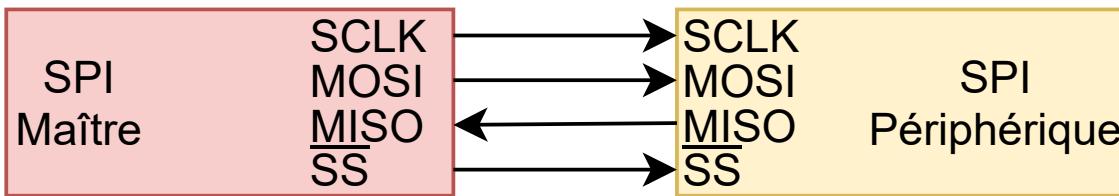


Figure 1.2: Exemple de configuration SPI pour une communication entre un maître et son périphérique.

Le maître contrôle la communication en fournissant une horloge quand il souhaite échanger des informations. Il est possible d'avoir plusieurs périphériques simultanément avec un même maître, qui sélectionnera avec lequel il souhaite communiquer, mais cette fonctionnalité n'est pas utilisé dans ce projet.

1.1.2 Circuit logique programmable - FPGA

La technologie Field Programmable Gate Array (FPGA), ou porte à logique programmable, est un type de circuit logique programmable très répandu en informatique embarquée. Elle permet d'avoir les avantages d'un circuit logique par rapport à un processeur, en termes de consommation, de vitesse d'exécution et de coût, mais offre en plus la possibilité d'être

reconfigurée (là où un circuit logique sera fixé). Contrairement à un processeur, qui va exécuter une suite d'instructions qui lui est donnée et qui est modifiable facilement, le FPGA est programmé pour une exécution particulière et ne peut pas passer d'une exécution à une autre aussi facilement. Il est composé de cellules logiques élémentaires et de bascules logiques librement connectables. Mais les avancées technologiques des dernières années l'ont rendu plus performant en rajoutant des modules de mémoire vive (BRAM) ou encore de traitement du signal (DSP).

À l'intérieur même du FPGA, il est possible de définir un processeur logiciel. C'est ce que le fabricant Xilinx a réalisé avec Microblaze. Ce module implémenté dans le FPGA permet de réaliser des programmes en C et d'interagir plus rapidement qu'avec un processeur physique extérieur au FPGA.

1.1.3 Microcontrôleur

Pour réaliser des tâches spécifiques, parfois avec des contraintes de taille ou de prix, on ne peut pas toujours se permettre d'utiliser un ordinateur personnel. Les microcontrôleurs donnent la possibilité d'intégrer les mêmes éléments de base que ces derniers, à savoir un processeur, de la mémoire vive et de la morte, des interfaces entrées/sorties, etc. Leurs spécialisations permettent ainsi d'optimiser l'efficacité énergétique, la vitesse d'exécution des tâches, mais aussi la fiabilité du temps d'exécution. Il existe cependant plusieurs gammes de produits, allant du tout public, avec Arduino ou Rapsberry Pi, au très spécifique, pour l'industrie comme les contrôleurs de moteur de voiture ou des téléphones portables. Il existe aussi des types de microcontrôleurs spécialisés non pas pour une application, mais pour un type de tâche, par exemple les Pynq, couplant des FPGA à des processeurs ARM, les Jetson de Nvidia proposant des GPU intégrés ou encore, des TPU avec le Google Coral.

1.1.4 Convertisseur Analogique-Numérique - ADC

Un convertisseur Analogique-Numérique, en anglais Analog to Digital Converter (ADC), est un dispositif électronique ayant pour but de convertir une grandeur analogique (ici un courant électrique) en une valeur numérique, pour qu'elle soit ensuite interprétable par d'autres systèmes. Dans ce projet, l'ADC utilisé est dit «front end», car il ne va pas seulement convertir le signal électrique reçu, mais il va aussi l'amplifier et le filtrer en partie pour le rendre exploitable, car l'EEG est un signal en microvolt d'une faible intensité.

1.1.5 Réseau de neurones artificiels

Inspiré du fonctionnement des neurones biologiques, le réseau de neurones artificiels est une méthode statistique optimisée par apprentissage, pour permettre de réaliser des tâches spécifiques. Souvent composés d'un ensemble de couches (on parle de réseau de plus en plus profond quand on en rajoute), avec une taille variable (au plus la couche est grande, au plus le réseau est large), ces réseaux sont utilisés pour de nombreuses tâches, comme la génération de visage, la détection des fraudes, la prise de décision hautes fréquences (pour des applications de trading par exemple), la détection de panneau en voiture, etc. En fonction de l'application, plusieurs types de réseaux peuvent être utilisés et combinés. Seuls ceux utilisés dans la suite de ce mémoire sont présentés :

- Le Fully Connected est la couche la plus basique, permettant à chacun de ses éléments de dépendre d'une combinaison linéaire des sorties de la couche précédente.
- Le Convolutional Neural Network (CNN) est un ensemble de couches de convolution. Il est bien souvent utilisé pour extraire des données pertinentes d'un vecteur (à une ou plusieurs dimensions), sans avoir besoin d'en connaître la position exacte. Il est très utilisé pour les images, mais aussi pour les signaux à une dimension.
- Le Recurrent Neural Network (RNN) permet d'utiliser des informations du passé pour mieux prédire l'avenir. En conservant un état caché qu'il va réutiliser d'une itération à l'autre, il peut garder en mémoire des informations pertinentes tout en oubliant ce qui ne devrait plus lui être utile. Il est plutôt utilisé dans les applications utilisant des données temporelles où le passé impacte sur l'avenir, comme les applications de trading ou la détection de motif dans un signal.

1.2 Éléments de la problématique

L'appareil développé dans cette maîtrise doit être capable de détecter des oscillations rapides dans l'activité cérébrale en temps réel, tout en étant portable et sur batterie. Une fois ces oscillations détectées, un stimulus doit être envoyé.

1.2.1 Acquisition d'un signal exploitable

Pour aller plus loin dans la stimulation en boucle fermée, il est nécessaire d'avoir un signal exploitable. En effet, toutes les prochaines étapes vont dépendre grandement de l'acquisition, car c'est l'unique source d'information. Le signal sortant des électrodes est soumis à des perturbations difficilement contrôlables. Des artefacts peuvent se créer à la suite de mouvement du sujet, d'augmentation du rythme cardiaque, d'un mauvais placement des électrodes ou encore à cause de perturbations électromagnétiques extérieures dues par exemple aux lignes à haute tension.

Pour tenter de répondre à ces problèmes, il est nécessaire d'appliquer des étapes de traitement du signal afin de le rendre le plus exploitable possible pour la suite.

1.2.2 Détection en temps réel d'oscillations rapides

Une fois le signal rendu utilisable, il faut en extraire les informations pertinentes. Sans même se focaliser sur l'étude de cas présentée chapitre 4, la détection de motif dans un signal en temps réel implique certaines contraintes. Déjà il n'est pas possible d'avoir accès à l'entièreté du signal, passé comme futur, ce qui est très utile pour certaines techniques comme les transformés de Fourier ou les décompositions en ondelette. Il faut se contenter de l'information contenue dans les quelques secondes précédant le signal actuel. De plus, il faut pouvoir répondre rapidement. Dépendamment de l'application, cela peut varier de quelques centaines de millisecondes à quelques secondes.

Dans le cas des «sleep spindles», il faut la détecter avant la fin de sa période, qui est inférieure à 2,5 secondes. Cependant, une meilleure précision temporelle est désirée, pour permettre d'étudier les impacts de la stimulation en fonction de la phase stimulée (on peut par exemple se demander si stimuler le début de la «sleep spindle» est-il plus intéressant que de stimuler le milieu ou la fin ?).

De plus, la stimulation en boucle fermée s'oppose à la boucle ouverte par une prise en considération de l'état présent. Une boucle ouverte sera plus aléatoire, en produisant des stimuli sans savoir ni pouvoir vérifier ce qui est vraiment stimulé. La boucle fermée s'oppose à cela en

utilisant l'état présent pour savoir si un stimulus doit être envoyé. La précision du détecteur, autrement dit la fiabilité pour détecter si le motif est bien présent, est donc importante. En effet, envoyer des stimuli qui ne stimulent pas le motif désiré se rapprocherait des performances d'un système en boucle ouverte.

Le temps réel et la précision sont les deux critères importants du détecteur désiré. Évidemment, détecter la majorité des motifs reste l'objectif principal.

1.2.3 Système embarqué fonctionnant sur batterie

Pour pouvoir utiliser cet appareil facilement dans plusieurs situations (en laboratoire ou chez le sujet) il est nécessaire qu'il soit portable. Son coût peut ainsi devenir plus abordable (mais cela dépend évidemment des technologies utilisées) et sa consommation de puissance peut être réduite (moins de composants, utilisation plus spécifique). Cette dernière est assez importante pour un fonctionnement sur batterie. Dans le cas de stimulation apparaissant pendant le sommeil, comme les «sleep spindles», il faut prévoir facilement dix heures de fonctionnement sur batterie.

L'utilisation d'un système embarqué rajoute des contraintes pour la puissance de calcul. En effet, vouloir exécuter des programmes le plus rapidement possible nécessite une plus grande puissance de calcul. La conception et l'optimisation de ces derniers sont nécessaires pour répondre à cette nouvelle contrainte sans perturber les autres.

1.2.4 Stimuler l'activité cérébrale

Il existe plusieurs solutions pour stimuler l'activité cérébrale, qui sont présentées dans le chapitre 4. Pour cette maîtrise, l'objectif est de réaliser une stimulation auditive ou donner la possibilité d'utiliser un autre appareil de stimulation. Pour que l'envoi de ce stimulus permette une stimulation, il doit être de l'ordre de quelques millisecondes et être le plus constant possible pour permettre la meilleure précision temporelle.

1.3 Objectifs de recherche

Le travail de recherche présenté dans ce mémoire et dans l'article chapitre 4 a pour but de répondre aux objectifs de recherche suivants :

1. Acquisition et traitement du signal EEG par un système embarqué.
2. Détection en temps réel des «sleep spindles» à l'aide d'un réseau de neurones.
3. Implémentation d'un réseau de neurones sur un système embarqué temps réel.

4. Stimulation en boucle fermée de l'activité cérébrale.
5. Rendre adaptable l'appareil à d'autres applications du domaine des neurosciences.

1.4 Plan du mémoire

La suite de ce mémoire est divisée en quatre sections. Chapitre 2 présente une revue de littérature. Chapitre 3 donne un aperçu de la méthode de recherche et de l'organisation du mémoire, en expliquant le lien entre l'article et les objectifs de recherche. Chapitre 4 présente un article qui introduit l'appareil développé pendant la maîtrise : un système embarqué de stimulation en boucle fermée de l'activité cérébrale, et y présente une étude de cas. Chapitre 5 propose une discussion générale sur les résultats obtenus dans le chapitre précédent. Chapitre 6 conclut en résumant l'ensemble du travail de recherche, ses limites et propose des pistes de recherches futures.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre est un aperçu de l'état de l'art du domaine de recherche étudié dans ce mémoire : l'acquisition du signal EEG, la stimulation en boucle fermée, le cas particulier de la détection de «sleep spindles», les réseaux de neurones en FPGA ainsi que l'optimisation multiobjectif. Une approche similaire est présentée dans le chapitre 4.

2.1 Acquisition du signal EEG

Le signal EEG est un signal électrique mesuré à l'aide d'électrodes placées sur la tête du sujet. Cette technique est non invasive : les électrodes sont placées à la surface du cuir chevelu. Cependant, il est possible de faire l'acquisition du signal au plus proche des neurones avec des électrodes intracrâniennes : l'iEEG. C'est par exemple utilisé par Zelmann *et al.* [2]. Mais c'est une installation trop invasive qui n'est pas voulue pour l'appareil introduit dans ce mémoire : le Portiloop. Un exemple de signal est présenté dans la Figure 2.1.

En fonction de leur utilisation, les systèmes d'acquisition peuvent avoir des prix dépassant le millier de dollars, car ils donnent la possibilité d'utiliser un grand nombre d'électrodes (supérieur à 64 pour certains), et de ne pas mesurer seulement l'EEG, mais aussi d'autres biosignaux comme la respiration, le rythme cardiaque, le mouvement des yeux, etc. Pourtant, toutes ces améliorations ne sont pas forcément nécessaire pour un système d'acquisition embarqué. La portabilité de ce dernier impose des choix des conceptions pour limiter les composants. Plusieurs recherches ont déjà proposé des systèmes embarqués permettant cette acquisition [3–6], en insistant même sur leur bas coût [7, 8].

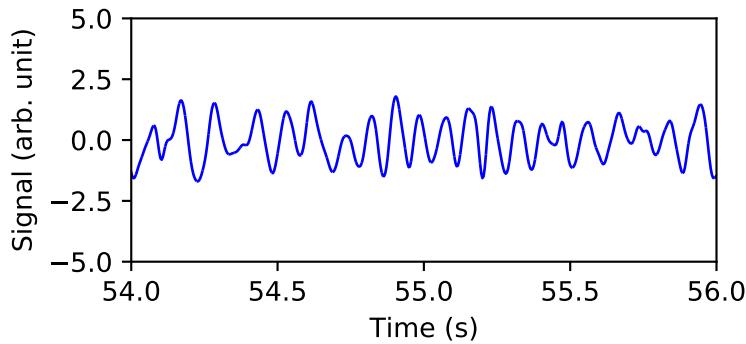


Figure 2.1: Signal EEG pendant le sommeil

2.2 Stimulation en boucle fermée

La stimulation en boucle fermée a déjà été étudiée dans de précédentes recherches [2–6,8–11].

Pour lutter contre la somnolence au volant d'une voiture, Lin *et al.* [4] propose un dispositif en deux parties. La première est un microcontrôleur embarqué qui s'accroche sur une casquette et qui va acquérir et traiter le signal. Il sera ensuite envoyé vers la deuxième partie : un ordinateur personnel. Loin d'être un système embarqué, ce dernier a pour rôle de récupérer le signal et d'en calculer la transformée de Fourier pour extraire des composantes pertinentes permettant, après utilisation d'un modèle linéaire, de déterminer le niveau de somnolence de l'utilisateur. Si nécessaire, le PC communique avec le microcontrôleur afin d'envoyer un signal sonore ou visuel. Cette solution date de 2008, ce qui explique en partie le besoin d'une unité centrale pour réaliser les calculs. Grâce à l'évolution rapide des technologies, il n'est plus nécessaire de déléguer ce genre de calcul. Si le projet était développé aujourd'hui, une autre approche aurait sûrement été adoptée. L'appareil présenté dans ce mémoire pourrait être adapté à cette application et permettrait un temps de réponse plus rapide, qui peut être vital pour l'utilisateur au volant de sa voiture.

Le bandeau et l'écharpe proposés par Kosmyna et Maes [6] ont pour objectif d'améliorer l'attention et l'engagement pendant des tâches passives, comme assister à une conférence ou à un cours magistral. Le bandeau fait l'acquisition du signal EEG et en extrait ses composantes fréquentielles. Elles sont calculées sur les 5 dernières secondes du signal, car le temps réel n'est pas critique ici. Si l'attention de l'utilisateur est trop faible, alors il reçoit un retour haptique par l'écharpe, qui va se mettre à vibrer. Il prendra alors conscience qu'il se décentre et pourra essayer de recentrer sa pensée. Le Portiloop pourrait à nouveau s'adapter à cette application, non pas en raison de sa vitesse d'exécution, mais plutôt pour la solution d'apprentissage profond qui permettrait peut-être une meilleure détection de la perte d'engagement.

Les attaques cardio-vasculaires pourraient être détectées et des stimulations auraient un effet thérapeutique sur le patient. C'est ce que propose von Lühmann *et al.* [5] avec leur microcontrôleur venant se placer sur la tête. Il fait l'acquisition du signal EEG, mais aussi ECG (electrocardiogramme, pour le rythme cardiaque), EMG (electromyogramme, pour l'activité des nerfs et des muscles) ainsi qu'une spectroscopie. En fusionnant toutes ces données, le système est capable de détecter une attaque cardio-vasculaire et d'envoyer une stimulation électrique. Il est vrai que le Portiloop a été conçu pour acquérir le signal EEG, mais il est tout à fait possible de s'adapter à d'autres biosignaux. Cependant, la spectroscopie ne fait pas partie de ses fonctionnalités et il n'est pas encore adapté pour être placé directement

sur le sujet, ce qui est nécessaire ici. Par contre, les technologies utilisées dans le Portiloop pourraient être réappliquées sur cet appareil, pour en améliorer les performances.

Zotou *et al.* [8] proposent eux aussi un appareil de stimulation auditive de «sleep spindles» en boucle fermée. Basé sur des technologies assez proches de celles utilisées dans le Portiloop, ce système réalise une acquisition en temps réel du signal EEG, soit par ADC, soit par un capteur Bluetooth (contenant aussi un ADC) qui transmet le signal. La volonté d'utiliser des composants Bluetooth est difficile à comprendre, car la stimulation audio n'est pas sans fil et le transfert d'information pourrait rajouter un délai non nécessaire. Une fois les données acquises, le signal est filtré pour calculer un seuil adaptable. Si le signal traité dépasse ce seuil, alors une «spindle» est détecté et une stimulation envoyée. Il semblerait que le délai entre la détection de la «spindle» et l'envoi audio soit d'environ 500 ms. Comme expliqué dans la prochaine section, cette détection n'est pas très précise et une solution basée sur un réseau de neurones artificiels a de meilleures performances. Le Portiloop obtient en effet de meilleures performances, à la fois en détection, mais aussi en temps de stimulation.

D'autres applications pour étudier la consolidation de la mémoire existent aussi [9–11]. Mais aucun de ces appareils ne propose d'implémentation basée sur un réseau de neurones artificiels, qui pourrait permettre à la fois une meilleure précision et surtout qui donnerait la possibilité de s'adapter à toutes les applications présentées dans ces articles, sous conditions d'avoir un jeu de données annotées, le tout en respectant les contraintes temps réel imposées par la boucle fermée.

2.3 Détection des « sleep spindles »

La détection de « sleep spindles » est une tâche difficile. Avant de vouloir la réaliser, il est nécessaire de pouvoir s'évaluer. Un jeu de donnée est beaucoup utilisé dans la littérature [12–15] : le *Montreal Archive of Sleep Study* (MASS) [16], dans lequel des annotations sur les « sleep spindles » sont fournies par deux experts. En général, les projets utilisant ces données considèrent que si une « sleep spindle » est identifiée par un expert, alors c'en est une. Sauf que cette hypothèse n'est pas justifiée, et que l'accord entre les deux experts est assez faible, comme le montre un f1-score de 0,54 [1]. Pour essayer d'obtenir un jeu de données plus fiables, sur lesquels les experts obtiennent un meilleur score, le *Massive Online Data Annotation* (MODA) [1] a été conçu. En augmentant le nombre d'experts (en moyenne 5 par « sleep spindle »), Lacourse *et al.* ont réannoté une portion de MASS afin d'obtenir un accord de 0,72 (le f1-score) entre les experts, face aux annotations finales. Pour arriver à ces annotations, une moyenne des scores des experts a d'abord été réalisée et un seuil a été défini afin que seulement les annotations supérieures à ce seuil soient considérées comme des « sleep

spindles ». Ces annotations sont fusionnées lorsque trop proches (<100 ms) et trop courtes (<300 ms), puis supprimées si trop courtes (<300 ms) ou trop longues ($>2,5$ ms). Par la suite, nous n'utiliserons que les résultats sur ce jeu de données comme ayant de l'intérêt. On pourra tout de même critiquer sa taille : il ne contient qu'environ 24 heures de sommeil.

Pour revenir aux algorithmes de détection, il en existe déjà plusieurs. On peut les classer en deux catégories. D'une part, il existe les détecteurs hors-ligne, qui peuvent accéder à l'entièreté du signal et qui n'ont aucune limite de temps de calcul [15, 17–24]. Ils sont en général basés sur des heuristiques calculées à l'aide de transformées de Fourier ou de décompositions en ondelettes appliquées à des segments importants du signal, qu'ils soient passés ou futurs. D'autre part, il existe les détecteurs en temps réel, qui n'ont accès qu'au signal présent et passé [2, 8, 12–14, 25, 26]. Cependant, ces derniers sont en théorie tous exécutables en temps réels, mais en pratique seulement [2, 8] ont été implémentés et testés en temps réel, sauf que ces appareils utilisent des heuristiques plus simplifiées que les détecteurs hors ligne et moins performantes que les réseaux de neurones.

La taille des autres architectures, surtout celle de Kulkarni *et al.* [12] que nous avons testé, est très loin d'être adaptée pour un système embarqué, et l'on pourrait même douter que leur utilisation soit possible avec un flux de données continu et constant. SpindleNet, introduit dans [12], semble être l'état de l'art des détecteurs pouvant devenir temps réel. Il propose d'utiliser trois données d'entrée pour le réseau de neurones : le signal brut (qui en réalité est déjà filtré dans les jeux de données qu'ils utilisent), l'enveloppe du signal filtré entre 9 et 16 Hz (fréquences où se situent les «sleep spindles») et un ratio de composantes fréquentielles (une comparaison entre les fréquences entre 2 et 8 Hz et celles entre 9 et 16 Hz). Les deux premières sont envoyées dans leur propre branche composée d'un CNN puis un RNN, et les résultats de ces deux branches sont concaténés avec la troisième entrée pour être envoyés dans une couche linéaire (fully connected). Ils ont indiqué que chacune des entrées permettait d'améliorer le score final, mais on peut douter de l'ablation qu'ils ont réalisée. En effet, s'ils ont supprimé complètement un CNN et un RNN pour savoir si l'enveloppe était utile, alors ils ont réduit la taille du réseau par quasiment 2. C'est assez logique que les résultats soient moins bons. Par contre, si l'on remplace une entrée par une autre (ce que nous avons fait pour l'étude d'ablation dans le Portiloop), alors la taille du réseau n'est pas diminuée, et l'on peut ainsi observer si une entrée était réellement importante. Nous ne sommes pas arrivés aux mêmes conclusions qu'eux, et nous pensons que l'enveloppe est en réalité déjà incluse dans le signal brut et que le réseau est capable d'en extraire les mêmes informations. De plus, ce réseau a été testé sur MASS. Les scores annoncés, bien que très bons en théorie, ne le seront peut-être pas sur MODA. De plus, son côté temps réel, même avec leur configuration assez puissante, est à remettre en question. Ils annoncent par exemple un temps d'inférence

de 6 ms pour le réseau, mais la fréquence d'échantillonnage du signal est de 250 Hz, ce qui signifie qu'un nouvel échantillon arrive toutes les 5 ms. Un délai est sûrement à prévoir du côté des filtres. Comme ce réseau n'est pas en accès libre, qu'il est trop gros, et qu'il a été testé sur MASS, nous ne l'utiliserons pas. C'est cependant une architecture similaire sera essayée dans le chapitre 4. Nous l'implémenterons au mieux de notre connaissance pour le tester sur MODA et pouvoir nous y comparer.

Un autre réseau, proposé par Tapia et Estévez [13], essaie deux approches un peu différentes. La première est d'envoyer le signal brut en entrée d'un réseau de neurones, composé lui aussi de CNN et de RNN. La deuxième est d'envoyer la décomposition en ondelette en entrée d'un réseau assez similaire. Cette dernière solution n'est pas envisageable en temps réel. L'article conclut finalement qu'elle n'apporte rien de particulier aux résultats et que cette piste de recherche n'est pas la plus prometteuse. De plus, ce réseau est aussi entraîné sur MASS, on ne peut donc pas se baser sur ces résultats.

2.4 Réseau de neurones en FPGA

Pour pouvoir utiliser ces réseaux de neurones en temps réel, l'utilisation du CPU du microcontrôleur n'est pas suffisante. Pour plusieurs raisons :

- L'acquisition est faite sur le FPGA, il faut donc transférer ces données vers le CPU, ce qui peut prendre du temps en fonction de la quantité de données.
- Les calculs ne sont pas assez parallélisés et le CPU n'est pas optimisé pour cette tâche.

Des projets ont été développés pour chercher à accélérer ces opérations à l'aide de FPGA. C'est le cas en particulier de Vohra *et al.* [27] introduisant « PYNQ-Torch », un framework permettant d'accélérer pytorch pour certaines utilisations. Pour cela, les auteurs ont implémenté certaines fonctions de pytorch en FPGA qui sont appelées par le framework en substitut des originales. Cependant, comme ils le soulignent à la fin de l'article, le transfert de données entre le CPU et le FPGA par Direct Memory Access (DMA, c'est à dire un accès direct à une zone de la mémoire partagé entre le CPU et le FPGA, où ils peuvent tous deux écrire et lire des informations) empêche finalement d'optimiser le temps d'exécution. En s'inspirant de leur solution, le Portiloop propose une solution pour implémenter intégralement le réseau de neurones en FPGA pour limiter grandement les échanges de données par DMA.

D'autres projets existent, comme [28] qui cherche à optimiser des opérations gourmandes en puissance de calcul et souvent utilisée en apprentissage profond, mais ce que l'on souhaite ici, c'est d'avoir un réseau complet en FPGA, pour réduire au maximum le temps d'inférence.

2.5 Optimisation multiobjectif

L'objectif premier d'un réseau de neurones artificiels est de maximiser ses performances. Cependant, dans la situation de la stimulation en boucle fermée, le temps d'inférence de ce réseau est aussi important. À cela, il faut ajouter des contraintes matérielles qui empêchent l'utilisation de trop gros réseaux. On peut transformer ce problème en une optimisation multiobjectif, que Yin *et al.* [29] ont essayé de résoudre avec leur méthode : *Sequential Model-Based Optimization* (SMBO). Elle consiste à rechercher la configuration optimale qui minimisera deux objectifs dans l'espace des hyperparamètres possibles d'un algorithme. Dans leur cas : la latence et la perte (dans la situation d'un réseau de neurones, sinon on pourrait minimiser l'inverse de la performance d'un autre algorithme). Comme son nom l'indique, cette méthode se réalise de manière séquentielle. Une grande quantité d'ensembles d'hyperparamètres sont échantillonnés autour du dernier réseau testé (avec une probabilité d'être échantillonné de manière complètement aléatoire aussi). Un réseau de neurones a été entraîné pour prédire la performance de l'algorithme testé. Ces ensembles sont envoyés vers le réseau pour qu'il prédisse leur perte probable. En parallèle de cela, des mesures de latence sont effectuées pour estimer une latence moyenne de chaque ensemble. Chaque ensemble d'hyperparamètres reçoit ensuite un score d'efficacité déterminé par sa probable position dans l'espace des configurations possibles (avec la perte et la latence, respectivement en ordonnée et en abscisse). Un ensemble aura un meilleur score s'il se trouve sous le front de Pareto, défini par les configurations qui ont une plus petite perte que toutes les configurations ayant une plus petite latence qu'elles. L'ensemble d'hyperparamètres avec la meilleure efficacité (parfois, un ensemble aléatoire) est ensuite sélectionné pour mesurer réellement sa perte. Il est ensuite ajouté au jeu de données du réseau de neurones qui prédit les pertes et à l'ensemble des configurations existantes. On recommence depuis la première étape jusqu'à trouver une configuration qui répond bien aux exigences.

Ce que l'on peut reprocher à cette méthode, c'est sa vitesse d'exécution. En effet, réaliser ces étapes séquentiellement peut être très problématique lorsque l'ensemble des hyperparamètres à tester prend du temps. Nous avons aussi remarqué que le réseau de prédiction n'arrivait pas à prédire avec précision au-delà d'une certaine limite, et que les configurations les plus petites étaient toujours sélectionnées, ce qui nous a obligés à rajouter des pénalités pour ces ensembles d'hyperparamètres. Dans le chapitre 4 une version parallélisée et améliorée de cette méthode est proposée.

CHAPITRE 3 MÉTHODE DE RECHERCHE ET ORGANISATION DU MÉMOIRE

Ce chapitre présente la méthode de recherche utilisée, en quoi elle répond aux objectifs définis dans la section 1.3, son lien avec l'article du chapitre 4 ainsi que l'organisation de ce mémoire. La méthode de recherche se divise en deux objectifs principaux : acquérir le signal pour le rendre utilisable et détecter le motif désiré pour prendre la décision de stimuler ou non. Dans ce mémoire c'est la détection de «sleep spindles» qui est étudiée.

3.1 Acquérir un signal exploitable

Pour pouvoir réaliser de la stimulation du cerveau en boucle fermée il faut pouvoir mesurer ce qu'il s'y passe. L'EEG est une technique qui permet d'acquérir l'activité cérébrale électrique de manière non invasive. Il existe des composants électroniques permettant de réaliser cette acquisition. Cependant, il est nécessaire d'appliquer des étapes supplémentaires de traitement du signal pour le rendre exploitable. Le chapitre 4 présente un appareil intégrant tous les composants permettant de réaliser ces étapes, tout en donnant la possibilité de les adapter en fonction du besoin de l'utilisateur. Ces étapes sont réalisées le plus rapidement possible afin de répondre à des contraintes temps réel imposées par la difficulté de la tâche demandée, qui peuvent imposer d'acquérir et traiter le signal en quelques dizaines de millisecondes.

3.2 Détection en temps réel des «sleep spindles»

L'objectif historique de ce mémoire est la stimulation auditive des «sleep spindles», mais le Portiloop offre la possibilité de s'adapter à d'autres applications. C'est pour cela que l'étude de cas présentée dans le chapitre 4 porte sur la stimulation de ces oscillations. La détection de ces dernières n'est pas une tâche facile, et rajouter une contrainte temps-réel, à savoir stimuler avant la fin de leur période n'aide pas. L'utilisation d'un réseau de neurones pour réaliser cette tâche avec le plus de précision possible est la solution retenue et implémentée. La recherche d'architecture et l'implémentation de ce dernier représentent les deux étapes majeures de cet objectif. Les deux travaillant de pair afin de toujours répondre à la nécessité de rendre l'appareil portable, sans jamais négliger le temps réel.

3.3 Structure du Document

Ce document suit la structure recommandée pour un mémoire par articles dans lequel la publication soumise est incluse sous la forme d'un chapitre.

- Chapitre 1 introduit le contexte de recherche et les concepts de bases.
- Chapitre 2 fournit un aperçu des contributions pertinentes de la littérature.
- Chapitre 3 présente la méthode de recherche.
- Chapitre 4 introduit un appareil embarqué de détection et de stimulation en temps réel de l'activité cérébrale. Cet article a été soumis à IEEE Transaction on Biomedical Circuits and Systems le 12 août 2021.
- Chapitre 5 propose une discussion générale sur les résultats présentés dans le chapitre 4.
- Chapitre 6 résume l'ensemble du travail, ses limites et les directions possibles pour de futures recherches.

CHAPITRE 4 ARTICLE 1 : THE PORTILOOP : A DEEP-LEARNING-BASED OPEN-SCIENCE TOOL FOR CLOSED-LOOP BRAIN STIMULATION

Preface :

Full Citation : Nicolas Valenchon¹, Yann Bouteiller¹, Hugo R. Jourde², Emily B.J. Coffey², and Giovanni Beltrame¹, “The Portiloop : a deep-learning-based open-science tool for closed-loop brain stimulation,” *IEEE Transaction on Biomedical Circuits and Systems*, 2021.

DOI :

Copyright : © 2021 IEEE. Reprinted, with permission from the authors.

Abstract - Electroencephalography (EEG) is a method of measuring the brain’s electrical activity, using non-invasive scalp electrodes. In this article, we propose the Portiloop, a deep learning-based portable and low-cost device enabling the neuroscience community to capture EEG, process it in real time, detect patterns of interest, and respond with precisely-timed stimulation. The core of the Portiloop is a System on Chip composed of an Analog to Digital Converter (ADC) and a Field-Programmable Gate Array (FPGA). After being converted to digital by the ADC, the EEG signal is processed in the FPGA. The FPGA contains an ad-hoc Artificial Neural Network (ANN) with convolutional and recurrent units, directly implemented in hardware. The output of the ANN is then used to trigger the user-defined feedback. We use the Portiloop to develop a real-time sleep spindle stimulating application, as a case study. Sleep spindles are a specific type of transient oscillation (~2.5 s, 12-16 Hz) that are observed in EEG recordings, and are related to memory consolidation during sleep. We tested the Portiloop’s capacity to detect and stimulate sleep spindles in real time using an existing database of EEG sleep recordings. With 71% for both precision and recall as compared with expert labels, the system is able to stimulate spindles within ~300 ms of their onset, enabling experimental manipulation of nearly the entire spindle. The Portiloop can be extended to detect and stimulate other neural events in EEG. It is fully available to the research community as an open science project³.

¹N. Valenchon, Y. Bouteiller and G. Beltrame are with the Department of Computer and Software Engineering, École Polytechnique de Montréal, Montreal, Canada {nicolas.valenchon,yann.bouteiller,giovanni.beltrame}@polymtl.ca

²H. Jourde and E. Coffey are with Department of Psychology, Concordia University, Montreal, Canada emily.coffey@concordia.ca, hjourde.clasp@gmail.com

³<https://github.com/mistlab/portiloop>

4.1 Introduction

Electrical activity within the brain forms the basis of perception, thought and behaviour. This endogenous electrical activity tends to be oscillatory in nature, as reciprocal connections within and between brain regions form functional circuits for processing and communicating information, and it can be measured on the scalp using electroencephalography (EEG). Correlational studies have been performed for nearly a century that attempt to link specific patterns and frequency bands in EEG to cognitive functions or brain states. These studies are informative and increased our understanding of brain processes. However, they are unable to establish causal relationships. The ability to interact with brain oscillations in a precisely-timed fashion to enhance or inhibit endogenous processes - using sensory [3,5,6,9], electrical [30] or magnetic [31] stimulation - allows for their functional roles to be determined [32], and potentially for restoration of processes deteriorated by aging or pathology [33]. While there is a great deal of interest in *closed-loop* stimulation [32,34], researchers lack flexible, powerful tools that are easily accessible. Research efforts are also limited by the portability of systems, and by their complexity and expense. In the current work, we develop the *Portiloop*, a complete, portable system for closed-loop stimulation. We demonstrate the Portiloop on a case study of scientific relevance : the auditory stimulation of sleep spindles.

Using auditory stimulation, researchers have enhanced *slow oscillations* (SOs), which are high amplitude waves (0.5 - 1.5 Hz) known to be involved in memory consolidation [9]. Ngo *et al.* showed SOs enhancement caused an overnight improvement in memory performance, a result that has now been replicated (see [35–37] for reviews). Slow oscillations are thought to work in concert with other faster oscillations, called *sleep spindles*, to reactivate recently learned memories and transfer them to long-term memory [38,39]. Sleep spindles are transient oscillations observed in both lighter and deeper non-rapid eye movement (NREM) sleep (*i.e.*, stages 2 and 3). Their role in memory consolidation is supported by increases in spindle density following learning (*e.g.*, [40]). Age-related changes in sleep spindles are correlated with differences in overnight performance gains (*e.g.*, [41,42] ; see [43] for a review of spindle mechanisms and functions).

Unfortunately, current systems that are capable of slow oscillation stimulation have more difficulty accurately and precisely detecting and stimulating sleep spindles in real time. Particular challenges of spindle stimulation are that each oscillatory cycle is only ~ 60 ms long and the entire spindle is between 0.5 and 2.5 s, leaving little time for traditional window-based frequency analysis ; there is considerable variability between the frequency, amplitude, and duration of individuals' spindles, particularly in older populations [44,45] ; and even for offline detection of spindles (which is an easier problem as the entire spindle is available and

can be used in detection), agreement on spindle identification between experts and algorithms is limited ($\sim 70\%$) [1, 17].

If it were possible to influence spindles with sound, as it is to enhance slow oscillations, researchers could explore their functional role in healthy adults as well as characterize their involvement in cognitive aging, and even perhaps restore degraded function. Although auditory stimulation is attractive due to its non-invasiveness, it is only one of several possibilities : specific brain areas can be more directly and forcefully stimulated using transcranial electrical or magnetic stimulation, which could also be triggered by the same device. Furthermore, by designing the system flexibly such that it can be extended to detect and stimulate brain oscillations other than spindles, we can greatly expand its application, for example to theta-band oscillations that are associated with working memory capacity and task performance [46].

The goal of this work is to design and explore the properties of a deep learning-based, portable, battery-efficient and low-cost device that will enable the neuroscience community to collect and process EEG data in real-time, detect patterns of interest for fundamental research questions, and respond at low latency with precisely-timed stimulation. We propose a neural network-based detection algorithm in FPGA and a parallelized design space exploration algorithm to optimize its parameters. We aim to accelerate fundamental research on closed-loop stimulation in neuroscience by creating a highly functional device and offering the code and plans to developers and scientists in the research community.

4.2 Related work

4.2.1 Devices for EEG acquisition and stimulation

Various portable devices have been developed to acquire and process EEG signals. In [7], the authors developed a low-cost device limited to acquisition. Other portable devices enable closed-loop stimulation [3, 5, 6], some also based on low-cost hardware [8], but work with simple heuristics and are generally not sufficiently powerful for deep learning applications. Closed-loop stimulation has been used in the context of preventing drowsiness [3], enhancing attention and engagement [6], preventing strokes [5] and studying memory consolidation [8–11]. Depending on the application, real-time constraints can vary from hundreds of ms [47] to a few seconds [6]. The Portiloop is the first system to provide a portable, real-time and deep learning-based solution for all these applications.

4.2.2 Offline sleep spindle detection

Machine learning-based detection algorithms require large sets of accurately labeled data. The consistent detection and labeling of sleep spindles is a challenging task, due to variability in their appearance and strength. Traditionally, spindles are visually identified by experts. One commonly used dataset [12, 13] is the Montreal Archive of Sleep Studies (MASS) [16], in which the sleep spindle annotations were provided by two experts. Projects using MASS for training usually take spindles identified by either expert. However, the MASS annotations have a low inter-rater agreement, as quantified by the f1-score⁴ of 0.54 [1]. The Massive Online Data Annotation (MODA) [1] project addressed this issue by having 5 experts (on average) annotate spindles and rate their confidence, in each EEG segment. The experts had an inter-rater f1-score of 0.72 with respect to the final MODA labels. This score is considerably better than the MASS equivalent, and the number of experts, the scoring and the post-processing steps enable final labels of much higher precision.

Several offline sleep spindle detectors have been developed and tested on MODA [17–23]. However, these are generally heuristics that compute Fourier transforms or wavelet decomposition on large portions of the signal, including segments that would be in the future considering an online application, and thus cannot be implemented for real-time detection.

4.2.3 Online sleep spindle detection

Online detectors (*i.e.*, attempting detection during signal acquisition) face more challenging conditions than offline detectors, due to the unavailability of “future” data points. For example, the duration of the spindle is not yet known, and it is one of the identifying criteria commonly used by experts. Some existing heuristics filter the signal, compute power features and rely on thresholds to perform detection [8]. Yet, these approaches exhibit poor f1-scores. Deep learning can also be leveraged to perform online sleep spindle detection. This is done by first training an artificial neural network offline through supervised learning to detect sleep spindles, and then feeding the incoming signal to the trained detector. Several such models have been trained in previous work [12, 13, 25, 26]. However, these works do not consider hardware constraints that are central for our purpose : they use large models that are often unable to run in real time even on high-end GPUs, which makes them inapplicable on the Portiloop device. Moreover, they are usually trained and tested on MASS [16] with an ‘OR’ operation performed on its labels, and therefore limited performance [1]. In this work, we design a Pareto-optimal neural architecture that performs best on the MODA dataset [1]

⁴The f1-score, widely used in statistics, is described in Equation 4.6.

while satisfying our hardware and timing constraints. We validate our architecture against the state-of-the-art SpindleNet [12], initially used with the MASS dataset. When both architectures are trained and tested on MODA, ours outperforms the baseline, on top of running in real time with our hardware.

4.3 Methods

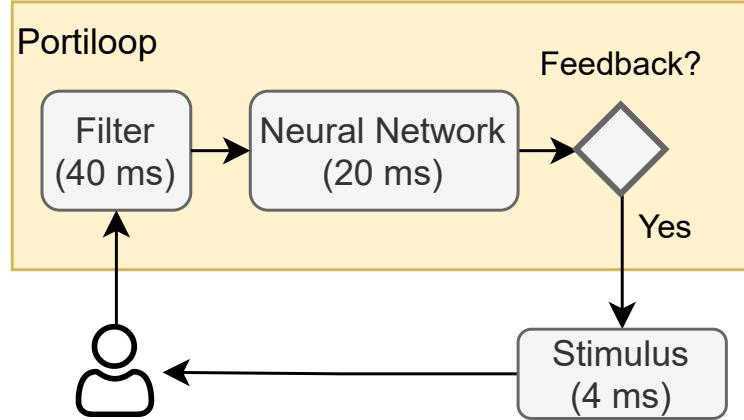


Figure 4.1: Main functionalities. Timing is reported from our case study. The constant delay in stimulation output (4 ms for auditory stimuli) enables precise experimental control stimulation timing relative to neural oscillatory phase.

A high-level description of the Portiloop system is provided in Figure 4.1. After being retrieved from the subject by electrodes and converted to numeric format by an Analog to Digital Converter (ADC), the EEG signal is processed through Finite Impulse Response (FIR) filters and fed to a trained Artificial Neural Network (ANN). The output of the ANN determines the nature of the feedback, *i.e.*, whether to stimulate the subject or not. These operations introduce their own constant delays in the closed-loop system (in Figure 4.1 ; note that the delays are measured on the sleep spindle detector that we derive in the second part of this article). Such delays are inherent to the order of the FIR filters, the architecture of the ANN and the nature of the stimulation. They have to be accounted for when building a Portiloop application, in order to satisfy the real-time constraints of the task.

4.3.1 Real-time constraints

To allow maximum flexibility in when the spindle can be stimulated, it must be accurately detected as early as possible following spindle onset. We identify two different sources of delay in the proposed system, *hardware* and *software* delays :

Hardware delays

By *hardware delays* we refer to the time it takes to retrieve the signal from the electrodes, convert it to digital, filter it, process it through the ANN, and send the resulting feedback stimulation to the subject.

Because we implement the Portiloop in Fast Programmable Gate Array (FPGA) logic, all these delays are constant. The hardware delays introduced by the electrodes, ADC and filtering are negligible in the Portiloop applications (*i.e.*, $\ll 1$ ms). The ANN delay depends on our implementation of the neural network architecture in FPGA. Since intra-layer operations can be parallelized in FPGA, it depends primarily on the depth of the ANN, *i.e.*, its number of layers. However, parallelization in FPGA has a cost in terms of resources, which are limited in the Portiloop. Thus, in practice, the width of reasonably large ANNs also has an impact on this delay. Finally, the stimulation delay depends on the nature of the stimulation. The Portiloop instantly sends a trigger from the ANN output, but this does not mean that the stimulus reaches the subject simultaneously, *e.g.*, 4 ms in the case of auditory stimulation.

Software delays

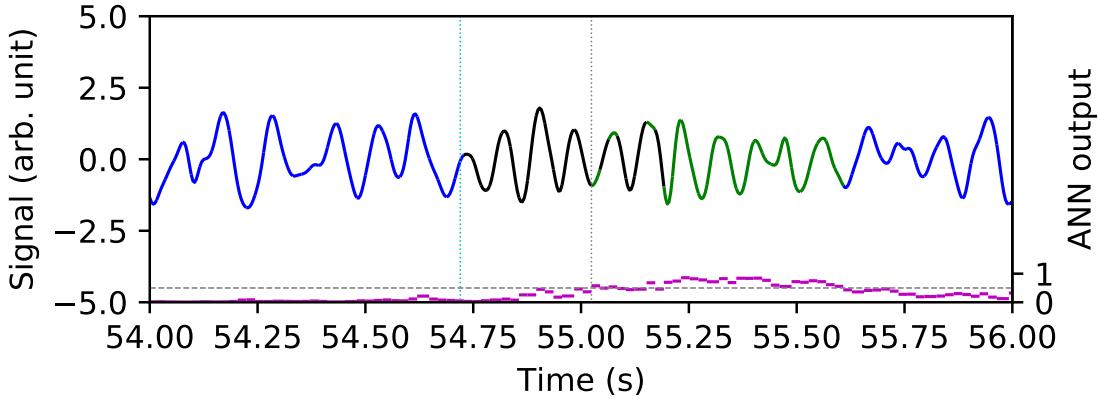


Figure 4.2: Software accumulation effect of recurrent units. Dark blue signal : no phenomenon of interest. Black signal : phenomenon of interest not detected by the ANN. Green signal ; phenomenon of interest correctly detected. Magenta : output of the ANN. Horizontal grey : detection threshold. The ANN introduces a software delay : the recurrent units of the ANN act as pseudo-accumulators, thus the detection happens with a variable delay.

Although the hardware operations performed by FIR filters are near-instantaneous, this type of filtering introduces a constant software delay in the filtered signal. This delay is a trade-off related to the order of the FIR filter. The higher the order of a FIR filter, the more efficient it is at filtering out undesirable frequencies, but also the longer the software delay introduced in

the signal by the filtering operation. More precisely, the delay d introduced by an FIR filter of order o on a signal sampled at frequency f is $d \propto \frac{o}{2f}$. These hardware and software delays sum to a constant total delay that is the minimum possible response time of the Portiloop system. Depending on the target timing constraints of the application, this leaves a certain margin for the actual detection of the phenomenon of interest in the filtered EEG signal. In practice, this detection may happen with its own variable software delay. An example of such delay is illustrated in Figure 4.2, where an ANN takes a variable amount of time before correctly detecting a transient pattern in an EEG signal.

4.3.2 Architecture of the proposed system

The detailed architecture of the Portiloop device is presented in Figure 4.3. The system is implemented on a Pynq Z2 [48] board. This board provides an integrated FPGA, which we use along the Vivado suite (v2019.1) to convert our software logic into hardware circuitry operating in real time.

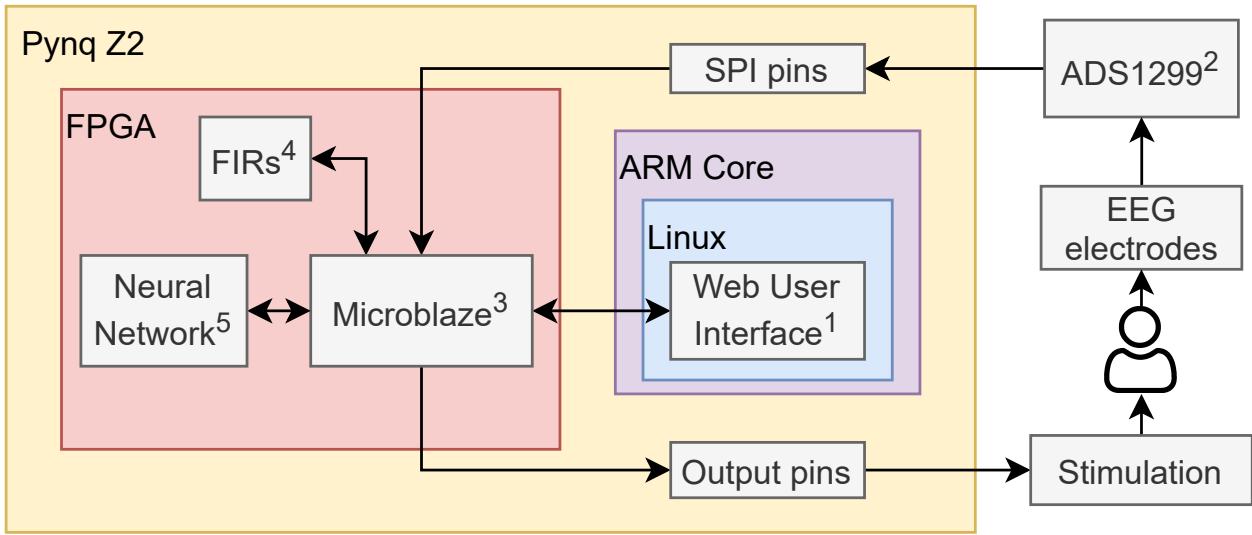


Figure 4.3: Detailed architecture of the Portiloop device. (1) The user controls the device from a simple Web User Interface. (2) An EEG front-end device acquires biosignals from the electrodes. (3) The Microblaze soft processor centralizes all operations performed within the FPGA. (4) The Microblaze core sends the digital signal to be filtered through Finite Impulse Response filters. (5) The Microblaze core sends the digital filtered signal to an artificial neural network. The neural network returns the likelihood of the input signal being the desired pattern in order to decide whether a stimulus should be sent.

Data acquisition

The analog signal captured by the electrodes is processed by a low-noise ADC front-end for EEG/ECG developed by Texas Instrument (ADS1299 [49]). This component can retrieve signal from 8 electrodes simultaneously, plus reference and bias electrodes. The ADC outputs a digital signal, data point per data point. Each data point is sent directly to the Pynq Z2 board at up to 16 kHz through Serial Peripheral Interface (SPI) communication.

Data preprocessing

Microblaze The Microblaze is a soft processor that is readily implemented by Xilinx on the FPGA. It acts as a central scheduler for all operations happening within the FPGA. We use the Microblaze to read data points from the SPI pins in real time and log them for possible post-analysis. We send either each or a relevant fraction of these data points to the FIR filters in real time by Direct Memory Access (DMA).

FIR filters We use FIR filters to preprocess the digital signal by filtering out undesired frequencies. These filters are readily available as Intellectual Property (IP) cores provided by Xilinx. We retrieve each unprocessed data point, filter it, and send the filtered data point back to the Microblaze through DMA. It is possible to implement as many FIR filters as needed, in the limit of the available space on the FPGA.

Additional preprocessing We can further preprocess the filtered signal directly within the Microblaze for simple operations. In particular, we standardize the signal on the fly though exponential moving average. In other words, we transform the filtered signal $s(t)$ to $s'(t)$ according to :

$$s'(t) = \frac{s(t) - \hat{\mu}(t)}{\hat{\sigma}(t)} \quad (4.1)$$

where $\hat{\mu}$ and $\hat{\sigma}$ are estimates of the mean and standard deviation of the filtered signal, computed as follows :

$$\delta(t) = s(t) - \hat{\mu}(t-1) \quad (4.2)$$

$$\hat{\mu}(t) = \hat{\mu}(t-1) + \alpha_\mu \delta(t) \quad (4.3)$$

$$\hat{\sigma}^2(t) = (1 - \alpha_\sigma)(\hat{\sigma}^2(t-1) + \alpha_\sigma \delta^2(t)) \quad (4.4)$$

$$\hat{\sigma}(t) = \sqrt{\hat{\sigma}^2(t)} \quad (4.5)$$

α_μ and α_σ being hyperparameters in $[0, 1]$. This custom real-time standardization makes the signal comparable from one subject to another, enabling generalizable learning.

Neural Network

Any real-time decision-making algorithm can be implemented on the Portiloop as long as it is written in FPGA-ready code. We focus on ANN models. The Portiloop being a portable, lightweight and battery-efficient system, it does not include any heavy computing units such as high-end GPUs. Instead, the ANN is synthesized as hardware circuitry in an FPGA, which provides only a limited amount of configurable circuitry. Thus, only small ANN models are allowed. We provide High-Level-Synthesis (HLS) C++ implementation of the following building blocks, based on [27] :

- 1D convolutional layers, used to learn and extract relevant patterns (*e.g.*, frequencies) in the signal.
- Layers of Gated Recurrent Units (GRUs), used to learn a memory representation, so as to keep track of the past. GRUs are usually on par with their famous Long-Short Term Memory (LSTM) cousin in terms of performance [50, 51], but they use roughly 30% less hardware resource and thus are more interesting in the context of FPGAs, where resources are limited.
- Fully connected layers, used to further increase the representational capacity of the model.
- Rectified Linear Unit (ReLU) non-linearities, which are simple $\max(x, 0)$ operations with minimal footprint.

We train an ANN offline using PyTorch⁵ , and once trained, we extract its weights from PyTorch and use our C++ library in the Vivado suite to convert the model (architecture and weights) into the FPGA. When a data point is fully preprocessed through the steps described in Section 4.3.2, the Microblaze sends a buffer of the last preprocessed data points by DMA to the ANN. In other words, we feed a sliding window of preprocessed signal as input to the model.

Inference is performed by the neural network in parallel to the acquisition and preprocessing of the next data point by the Microblaze. Once inference is done, the result is sent back to the Microblaze through DMA. Depending on this result, the Microblaze writes a code (*e.g.*, a trigger signal) on the output pins of the Portiloop device. Any stimulation device can be used with an appropriate adapter for the trigger signal.

⁵<https://pytorch.org/>

4.3.3 Type of ANN

The Portiloop system is designed to process EEG signals in real time. This type of input involves time-series of data containing oscillatory and transitory elements. In the realm of deep learning, a natural way of processing such data is to use either 1D convolutions, recurrent units, or a combination of both. The type of ANN architecture that we recommend is inspired by the SpindleNet [12] architecture. In essence, a sliding window over a few last data points is fed to a Convolutional Neural Network (CNN) whose purpose is to extract relevant features (*e.g.*, frequencies) in this signal fragment. Then, these extracted features are fed to a Recurrent Neural Network (RNN) whose purpose is to keep track of the features extracted in past forward passes. Note that another family of architectures, called Transformers [52], is known for exhibiting good results task-wise with this type of data. However, Transformers are memory-less and not suitable for lightweight real-time applications, because they need to process the whole signal at each forward pass. Conversely, RNNs are able to process one single data point at each forward pass and keep track of the past in memory, which makes them more relevant in Portiloop applications.

4.3.4 Pareto-optimal architecture search

The FPGA used in the Portiloop has a limited amount of available circuitry, so as to ensure its portability and low price. Thus, typically large ANN architectures such as SpindleNet [12] are orders of magnitude too large to be implemented in our device. Fortunately, as we illustrate in Section 4.4, those models are often much too large for their actual purpose. When developing a novel Portiloop application, one needs to devise a neural network that is both high-performance and lightweight, by selecting the right set of hyperparameters H in the space of all possible hyperparameter sets \mathcal{H} . Such hyperparameters include the size of the sliding window, the number of layers in each part of the ANN, the width of each layer, the time dilation (see Section 4.3.5), the type of optimizer, the hyperparameters of the optimizer itself, etc. \mathcal{H} can be very large, and finding a set of hyperparameters that yields a high-performance model within given hardware constraints is far from trivial. We introduce “Parallel Model-Based Optimization” (PMBO), a network-based algorithm that automates this process in a parallel fashion. Released as open-source along with our code, PMBO is essentially a parallelized and evolved version of “Probabilistic SMBO” [29]. PMBO is a guided-search approach that finds a suitable set of hyperparameters rapidly. For this matter, it uses one machine (or process) to train a *meta network* whose role is to predict non-trivially available costs for any given set of hyperparameters. Furthermore, PMBO uses any available machines (or processes) in parallel to train ANNs from sets of hyperparameters

selected based on the cost estimated by the meta model.

Software and hardware costs

The purpose of PMBO is to find Pareto-optimal sets of hyperparameters that minimize both a *software cost* and a *hardware cost*. Hyperparameter selection is a bi-objective problem in our setting :

- On the one hand, we want an ANN that performs well at detecting the desired patterns. We measure this performance in terms of the f1-score of our model. The f1-score depends both on the precision (how sure we are that positive outputs are true positives) and on the recall (how sure we are that we capture all true positives) of the model. It is defined as :

$$f1 = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}} \quad (4.6)$$

where

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (4.7)$$

and

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (4.8)$$

The f1-score is in $[0, 1]$, with 1 being a perfect classifier. We cast this into a minimization problem by defining our *software cost* as $L_s = 1 - f1$.

- On the other hand, we want our ANN to be as lightweight as possible, so it fits in the FPGA and executes as fast as possible. A precise measurement of the execution duration and amount of circuitry needed on the board is difficult. In fact, we do not have access to these results until the model is actually synthesized on the board, which is a long process. Thus, we use the number of trainable weights in the ANN as a proxy for these concerns, and call this number our *hardware cost* L_h . This cost is computed directly in PyTorch. Note that our choice of costs is arbitrary and other custom costs can be used instead.

Pareto efficiency

Let us consider a meta dataset \mathcal{E} of previously completed experiments (*i.e.*, tuples $E = (H, L_s, L_h)$ of hyperparameter sets with their real costs). Let us also consider the following Pareto-domination relation :

Definition 1. Let $E = (H, L_s, L_h)$ and $E' = (H', L'_s, L'_h)$ be two experiments in \mathcal{E} . We say that E Pareto-dominates E' and we denote $E < E'$ when $L_s < L'_s \wedge L_h < L'_h$. (NB : in the

context of this minimization problem, dominating means having the smallest costs, hence the notation)

For a given experiment E and meta dataset \mathcal{E} , we denote $D_{\mathcal{E}}(E)$ as the number of experiments in \mathcal{E} that are Pareto-dominated by E , and $d_{\mathcal{E}}(E)$ as the number of experiments in \mathcal{E} that Pareto-dominate E . In other words :

$$D_{\mathcal{E}}(E) = |\{E^i \in \mathcal{E}, E^i > E\}| \quad (4.9)$$

$$d_{\mathcal{E}}(E) = |\{E^i \in \mathcal{E}, E^i < E\}| \quad (4.10)$$

For a given set of hyperparameters $H \in \mathcal{H}$, we further define an estimate of the corresponding completed experiment E as $\hat{E} = (H, \hat{L}_s, \hat{L}_h)$, where \hat{L}_s and \hat{L}_h are estimates of the real costs, computed by the meta network. Note that in our setting, $\hat{L}_h = L_h$ is available and thus only \hat{L}_s is estimated by the meta network. We propose the heuristic Pareto efficiency $\eta(\hat{E})$:

$$\eta(\hat{E}) = a_{\mathcal{E}}(\hat{E}) + b_{\mathcal{E}}(\hat{E}) + s_{\mathcal{E}}(\hat{E}) - h_{\mathcal{E}}(\hat{E}) \quad (4.11)$$

where $a_{\mathcal{E}}(\hat{E})$ promotes hyperparameter sets whose predicted costs are not dominated by many experiments in \mathcal{E} :

$$a_{\mathcal{E}}(\hat{E}) = 1 - \frac{d_{\mathcal{E}}(\hat{E})}{|\mathcal{E}|} \quad (4.12)$$

$b_{\mathcal{E}}(\hat{E})$ promotes hyperparameter sets whose predicted costs dominate many experiments in \mathcal{E} :

$$b_{\mathcal{E}}(\hat{E}) = \frac{D_{\mathcal{E}}(\hat{E})}{|\mathcal{E}|} \quad (4.13)$$

$s_{\mathcal{E}}(\hat{E})$ promotes hyperparameter sets whose predicted software costs are better than the best software cost amongst all completed experiments in \mathcal{E} :

$$s_{\mathcal{E}}(\hat{E}) = \frac{\min(\{L_s, (H, L_s, L_h) \in \mathcal{E}\})}{\hat{L}_s} \quad (4.14)$$

and $h_{\mathcal{E}}(\hat{E})$ penalizes hyperparameter sets that have a high density with respect to experiments present in \mathcal{E} in terms of their hardware cost. More precisely, we define a range of hardware costs we are interested in, and we split this range into a number of bins. We then compute the binned density of experiments in \mathcal{E} over this range, and multiply this density by the range's width. The penalty $h_{\mathcal{E}}(\hat{E})$ is the height of the resulting bin where \hat{L}_h stands. Multiplying the density by the range's width enforces $h_{\mathcal{E}}(\hat{E}) > 1$ in regions of high density and $h_{\mathcal{E}}(\hat{E}) < 1$ in regions of low density.

Parallel Model-Based Optimization

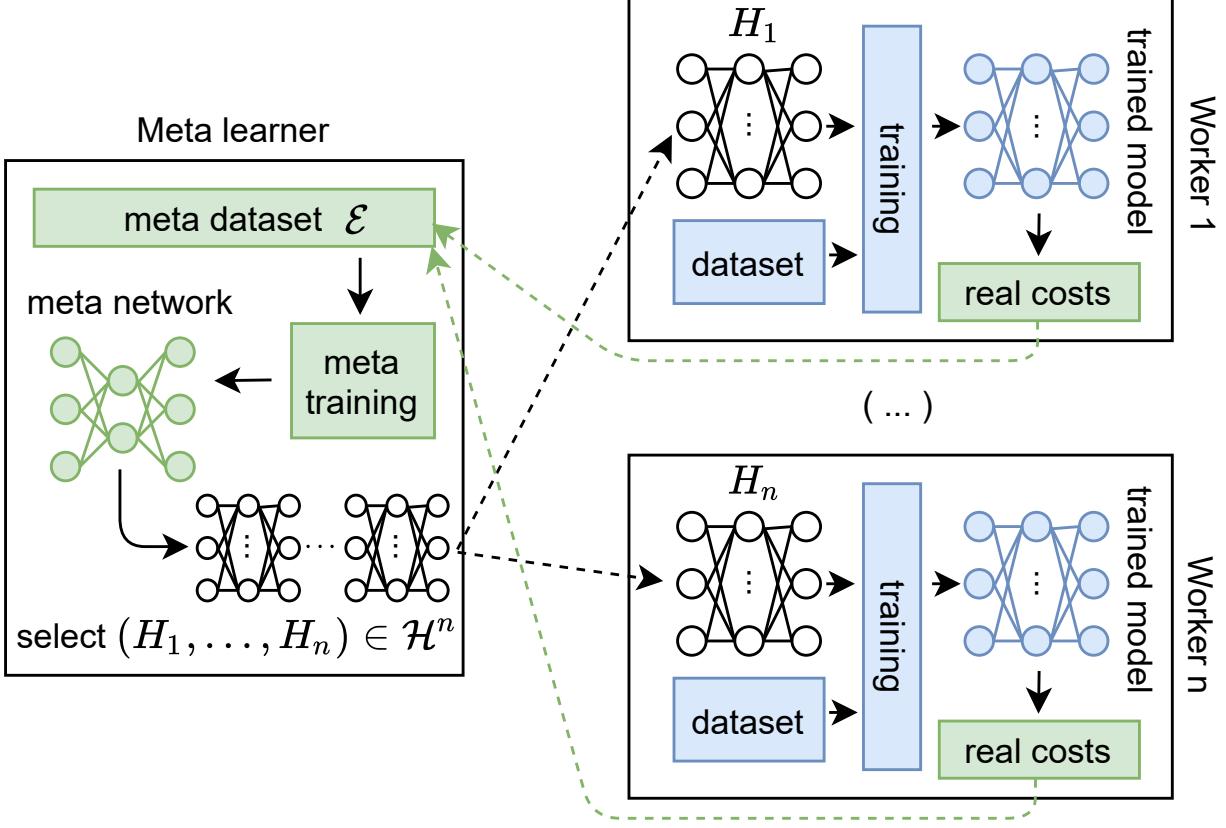


Figure 4.4: PMBO. The algorithm is based on a single producer and multiple consumers architecture. The meta learner is in charge of producing relevant hyperparameter sets in a guided fashion. Then, it sends them to idle workers, and keeps producing new sets as long as idle workers remain. Each worker that has received a new set starts training the corresponding ANN. Once this training ends, the real costs of the hyperparameter set can be computed and are sent back to the meta learner.

Figure 4.4 explains the PMBO algorithm. A central meta learner is communicating with n peripheral workers to find a hyperparameter set $H^* \in \mathcal{H}$ that is Pareto-optimal for both the software and hardware costs (*i.e.*, non-Pareto-dominated by any other set). For this matter, the algorithm uses a meta dataset \mathcal{E} of tuples $E^i = (H^i, L_s^i, L_h^i)$ to train a meta network that maps any hyperparameter set $H \in \mathcal{H}$ to its corresponding (estimated) costs \hat{L}_s and \hat{L}_h^6 . Once the meta network is trained, it is used to guide the sampling process. More exactly, we sample m hyperparameter sets in \mathcal{H} from a multivariate Gaussian distribution around

⁶ \hat{L}_h is an output of the meta network in the general case. However, with our choice of hardware cost it is not, since the ground truth L_h is available.

the hyperparameter set corresponding to the last results received from the workers by the meta learner. Sets that don't satisfy user-defined constraints (*e.g.*, that have already been tested, or, when L_h is available, that fall outside the range we are interested in) are discarded and resampled. We then select the best set in terms of Pareto efficiency, estimated thanks to the trained meta network. This selected set is appended to a bounded buffer, waiting to be consumed by an idle worker. The sampling process is repeated until the bounded buffer is full. When a worker is idle, it fetches an ANN architecture and training instructions from the bounded buffer. The worker yields a measurement of the real hardware and software costs for the current hyperparameter set, which are sent back to the meta learner and appended to the meta dataset. The meta learner then uses the updated meta dataset to train a new meta network, and so on.

4.3.5 Virtual ANN parallelization with time dilation

Time dilation [53] is a technique that enables recurrent units such as GRUs to look further back in time before gradients vanish, at no computational cost. We propose a version of this technique that allows us to virtually parallelize a single physical ANN into several decoupled virtual models. Our approach enables shallow recurrent neural networks to look further back in time by skipping the redundant information that is inherent to the use of a sliding window as input, while still acting as fast as possible. Figure 4.5 shows how time dilation can be used to look further back in time and avoid redundancy.

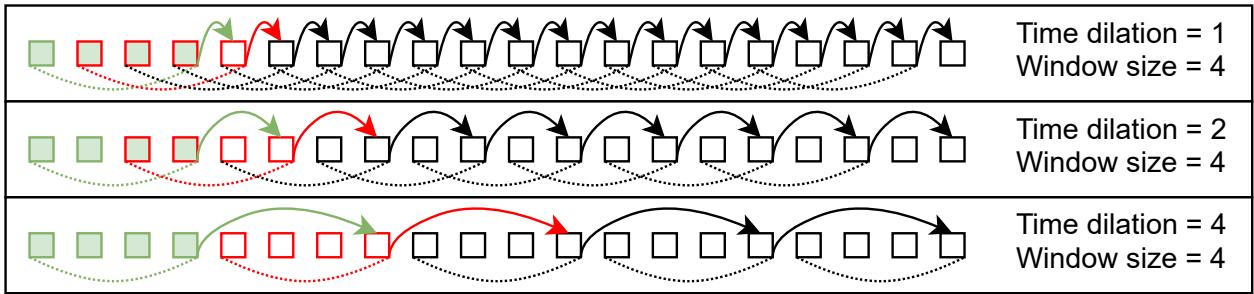


Figure 4.5: Time dilation. In this example, a sliding window of the 4 last samples (underlined with a dotted curve) is used as input to the model. The time dilation is the number of samples between two forward passes in the ANN. When it is small (top), two consecutive windows overlap for the most part (see *e.g.*, green and red windows). This implies that the recurrent hidden state which persists from one forward pass to the next (arrows) contains a lot of redundancy with the next input. When the time dilation is big (bottom), this issue is corrected, and back-propagation will reach much further back in time for the same number of forward passes. NB : forward passes happen only at the end of arrows in this diagram, the ANN is idle during other time-steps.

Although time dilation enables reaching further back in time at no extra computational cost, this comes with a cost in terms of delays. Since our technique causes samples⁷ to be skipped between forward passes in the ANN, a detection delay that can be as long as the time dilation is introduced. We correct this issue by implementing a trick that we call *virtual parallelization*. We create a First In First Out (FIFO) list as big as the time dilation, and fill this list with independent hidden states. At each time-step, we pop a hidden state from this list, feed it to the recurrent units of our ANN, perform a forward pass, and append the resulting hidden state to the list. Doing this without skipping samples is equivalent to having several decoupled models running in parallel as illustrated in Figure 4.6, although one single ANN is physically used.

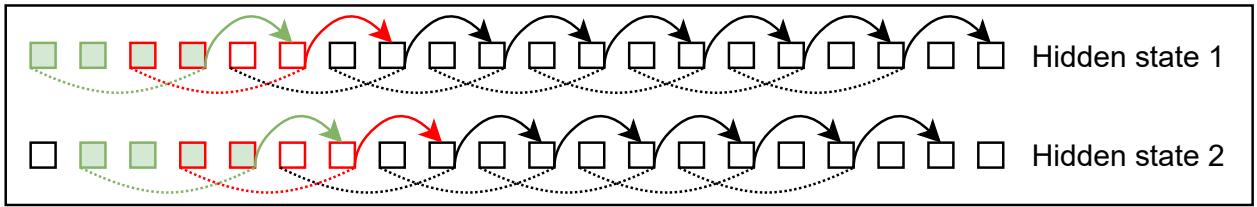


Figure 4.6: Virtual parallelization. In this example, the time dilation is 2. Thus, we keep track of 2 independent hidden states and feed these alternately to the recurrent units of the ANN. This is equivalent to having two decoupled models that are used alternately for inference.

This trick allows us to keep acting as fast as possible since it removes the need for skipping samples, while still reaching far back in time at no extra computational cost.

4.3.6 Case study : sleep spindle auditory stimulation

We now demonstrate the Portiloop implementation of stimulating the brain with sound during sleep spindles. The long-term goal of this application is to further clarify the role of sleep spindles in learning and memory, and to explore therapeutic interventions for memory decline. Stimulating sleep spindles is more challenging than low frequency neural activity like slow oscillations, due to their high frequency (\sim 12 to 16 Hz), rapid evolution (<2.5s), and therefore tight timing constraints. We propose the first portable device able to stimulate sleep spindles in real time with high detection performance. Because maximum experimental flexibility is attained by being able to stimulate at anytime during the course of the spindle including with phase precision, we conduct a thorough time analysis of the proposed system, with possible trade-offs to maximize specific performance.

⁷Here, by *sample* we refer to all data points acquired while a forward pass is executed in the ANN. For the sake of simplicity, our figures assume that there is only one data point per sample.

Neural architecture

To the best of our knowledge, the state-of-the-art in previous work regarding online sleep spindles detection was the SpindleNet [12] model. This architecture is however much too large to be implemented on the Portiloop system. Moreover, it is trained and evaluated on the MASS labels, which has been annotated by experts whose spindle evaluation varies considerably, and it is closed-source. Nevertheless, we draw inspiration from their work as a starting point for our ANN architecture design. In particular, we train models based on the same idea of using CNNs followed by RNNs, and we evaluate the relevance of the three different inputs used by SpindleNet (namely, the raw signal, the signal envelope and the power features) in our setting. We use PMBO along with the MODA dataset to derive a much smaller architecture, and provide a quantitative comparison with the SpindleNet architecture on MODA. Since we do not have access to the SpindleNet model, we rebuild the architecture described in [12] and train it on the MODA dataset with the same pipeline that we use to train our models.

Dataset and training

We use the MODA dataset, with ethics approval, for training our ANN, since its labels are considerably more reliable than *e.g.*, performing an ‘OR’ operation on the MASS labels [1]. This dataset is divided in two subsets. The first one, called *phase 1*, consists of 100 younger subjects, whereas the second one, *phase 2*, consists of 80 older subjects. The MODA dataset provides two types of annotations (labels) on the signal : the first is the mean score given by the group of experts for each data point ; the second is a binary classification of each data point as a spindle or non-spindle, defined by a threshold on the aforementioned scores (0.2 for phase 1 and 0.35 for phase 2). Further post-processing steps were applied to obtain these binary labels : spindles that were too short (< 0.3 s) and too close (< 0.1 s) to each other were merged, then spindles that were too short (< 0.3 s) or too long (> 2.5 s) were relabeled as negative. Given this dataset, two types of ANNs are possible : classifiers and regressors. These two types of ANNs differ only by the labels and losses used to train them. Classifiers are trained on the binary labels, by optimizing the binary cross entropy loss. They directly predict whether the current signal is a spindle or not, according to the very specific definition given by these binary labels (*i.e.*, taking into account the thresholds and post-processing applied by MODA). Regressors are trained on the score labels, by optimizing the mean square error loss. They predict the score given by the experts (before the aforementioned post-processing steps), which allows the user to select their own threshold for detection. Note that, in practice, classifiers also enable the user to select their own threshold,

although in a less interpretable way. We experiment with both types of models. Finally, note that MODA is a highly unbalanced dataset as only about 5 % of the signal is labeled as sleep spindles. During the course of this work, we tried different ways of balancing training for classifiers and regressors. Interestingly, we found that classifiers highly benefit from oversampling (*i.e.*, sampling 50 % of spindles and 50 % of non-spindles from the dataset during training) whereas all the balancing techniques we tried for regression (including oversampling, Label Distribution Smoothing [54] and a custom version of the latter) actually hinder the training.

Signal processing pipeline

Because we use the SpindleNet architecture as a baseline to evaluate our approach, we need to compute all three inputs used by this model *i.e.*, a clean signal, an envelope of the signal, and a “power feature ratio” [12]. The latter compares frequencies between 2 Hz and 8 Hz with frequencies between 9 Hz and 16 Hz from the Fourier transform over the last 500 ms of signal. Computing this ratio is resource-intensive in the context of the Portiloop system, plus we found that this did not help our models. Therefore we compute this ratio offline for the sole purpose of comparison with the SpindleNet architecture ; it is not used in our model. We set the ADC frequency to 500 Hz, which allows the Microblaze soft processor to log the raw signal at a finer-grained resolution than what we actually use for detection. The raw signal is then downsampled to 250 Hz. Figure 4.7 depicts the FPGA pipeline that computes cleaned signal and envelope.

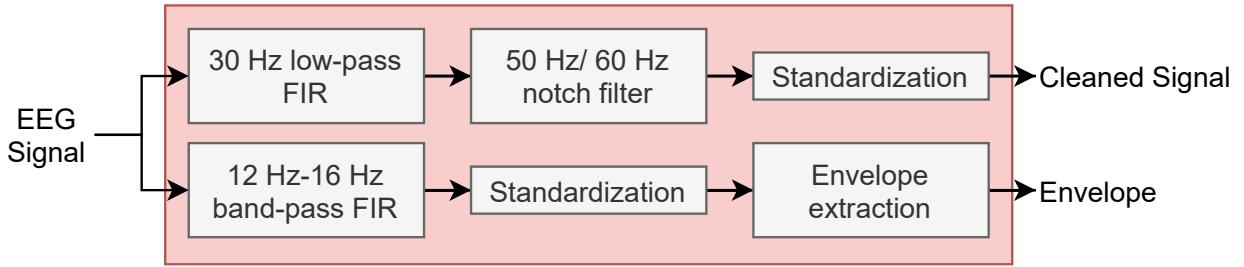


Figure 4.7: Signal processing pipeline extracting relevant inputs for the ANN. Power features are computed offline only for the SpindleNet architecture and are not represented in this diagram.

Since FIR filters introduce software delays, we designed both branches of the pipeline so that they introduce identical software delays to their respective outputs (*i.e.*, 40 ms at 250 Hz sampling rate), see Figure 4.7. We used the same frequency band to prepare our ‘cleaned signal’ in the first step of the pipeline as is used in standard sleep scoring (*i.e.*, 0.5 Hz to 30 Hz) [55].

An FIR filter of order 10 works reasonably well to remove frequencies above 30 Hz, but we observed persistent power line noise. To address this issue, we apply a notch filter whose frequency depends on the geographical area (50 Hz in Europe, 60 Hz in North America). For removing low frequencies, we rely on the online standardization process described in Section 4.3.2. We use a coefficient $\alpha_\mu = 0.1$ for our running average, which can be shown to attenuate frequencies under 4 Hz. We use a smaller coefficient $\alpha_\sigma = 0.001$ for our running variance, meaning that our estimate of the standard deviation takes a larger portion of the signal into account. We empirically found this choice of α_μ and α_σ to reveal EEG features of interest and yield sensible standardization, by visual inspection. We apply a similar procedure in the second branch to extract the envelope. First, the signal is filtered with a FIR band-pass between 12 Hz and 16 Hz. Then, it is standardized like in the first branch, except that we use $\alpha_\mu = \alpha_\sigma = 0.001$. We then square the signal and smooth the result by computing its moving average according to Equation 4.3, this time with $\alpha_\mu = 0.01$. This method is much simpler and faster than computing *e.g.*, an Hilbert envelope. We evaluate different types of ANN architectures, using either both or only one of these preprocessed signals as input.

Stimulation

The output of the ANN tells whether the model considers the current signal being a sleep spindle or not. Some further processing is necessary to ensure that we only send one stimulation per spindle. As seen in Figure 4.2, the detection can be noisy around the beginning or the end of a spindle, especially since we use decoupled virtual parallel networks (see Section 4.3.5). A stimulation is sent upon initial spindle detection. To avoid multiple stimulations of the same spindle the subsequent stimulation may only occur once the current stimulus ended and at least 400 ms following the ending of the spindle. If a spindle is detected again within this duration the timer is reset, since we consider it as being part of the previous spindle.

4.4 Experiments

We devise a Pareto-optimal neural architecture using the PMBO algorithm. We conduct a thorough quantitative and qualitative study of the resulting system, not only in terms of ethereal detection scores as generally seen in previous work, but also in terms of real-time stimulation performance. Note that all our experiments are based on the MODA dataset rather than actual nights spent wearing the Portiloop device, as we do not have ground truth labels for such data. Using the final device would require reproducing the experimental setting of MASS/MODA, which we leave to experts for future work.

4.4.1 Online detection

All results regarding online detection performance are summarized in Table 4.1. This table shows the f1, precision and recall metrics that statistically describe how efficient different models are at detecting sleep spindles (on average over all data points). These metrics are provided separately for phase 1, which groups younger subjects, for phase 2, which groups older subjects, and for the whole cohort.

Table 4.1: Quantitative results. The nomenclature is “mean (std)” and the superscripts are for referencing rows in the text.

	(a) Phase 1 (younger)			(b) Phase 2 (older)			(c) Whole Cohort		
	Recall	Precision	f1	Recall	Precision	f1	Recall	Precision	f1
Experts									
IExp ¹	0.76 (0.16)	0.81 (0.17)	0.76 (0.1)	0.66 (0.19)	0.74 (0.17)	0.65 (0.12)	0.72 (0.18)	0.78 (0.17)	0.72 (0.12)
Offline Detection									
<i>Ferrarelli [18]</i>	0.19	0.83	0.31	0.16	0.87	0.27	0.18	0.85	0.29
<i>Mölle [19]</i>	0.83	0.47	0.6	0.78	0.44	0.56	0.81	0.46	0.58
<i>Martin [20]</i>	0.61	0.64	0.62	0.58	0.56	0.57	0.6	0.6	0.6
<i>Wamsley [21]</i>	0.57	0.69	0.63	0.56	0.62	0.59	0.57	0.66	0.61
<i>Lacourse [17]</i>	0.75	0.73	0.74	0.7	0.69	0.7	0.73	0.71	0.72
<i>Ray [22]</i>	0.73	0.47	0.57	0.75	0.32	0.45	0.74	0.4	0.51
<i>Parekh [23]</i>	0.85	0.61	0.71	0.74	0.68	0.71	0.8	0.65	0.71
Online Detection									
<i>Based on SpindleNet [12]²</i>	0.92 (0.04)	0.24 (0.07)	0.38 (0.07)	0.85 (0.06)	0.19 (0.08)	0.3 (0.1)	0.89 (0.05)	0.22 (0.07)	0.35 (0.08)
2-input ³	0.68 (0.04)	0.6 (0.06)	0.64 (0.03)	0.52 (0.09)	0.58 (0.04)	0.54 (0.05)	0.62 (0.06)	0.6 (0.05)	0.61 (0.03)
2-input ablation 1 ⁴	0.7 (0.09)	0.47 (0.08)	0.55 (0.04)	0.56 (0.11)	0.43 (0.09)	0.47 (0.04)	0.65 (0.1)	0.46 (0.08)	0.52 (0.04)
2-input ablation 2 ⁵	0.72 (0.03)	0.57 (0.06)	0.64 (0.03)	0.57 (0.08)	0.53 (0.04)	0.55 (0.04)	0.67 (0.04)	0.56 (0.05)	0.61 (0.03)
1-input ⁶	0.7 (0.04)	0.59 (0.05)	0.64 (0.03)	0.54 (0.09)	0.58 (0.05)	0.55 (0.05)	0.64 (0.05)	0.59 (0.05)	0.61 (0.03)
1-input ablation td ⁷	0.47 (0.1)	0.6 (0.09)	0.51 (0.03)	0.31 (0.12)	0.59 (0.08)	0.39 (0.08)	0.41 (0.1)	0.6 (0.09)	0.47 (0.04)
1-input trained on p1 ⁸	0.72 (0.05)	0.56 (0.05)	0.63 (0.03)	0.57 (0.08)	0.52 (0.07)	0.54 (0.05)	0.66 (0.07)	0.55 (0.05)	0.6 (0.03)
1-input trained on p2 ⁹	0.75 (0.05)	0.5 (0.05)	0.6 (0.02)	0.62 (0.09)	0.45 (0.05)	0.52 (0.03)	0.7 (0.06)	0.49 (0.04)	0.57 (0.02)
1-input regression ¹⁰	0.62 (0.07)	0.64 (0.06)	0.63 (0.03)	0.53 (0.06)	0.55 (0.08)	0.53 (0.04)	0.58 (0.06)	0.62 (0.06)	0.6 (0.03)

Experts and offline detectors

As previously highlighted, sleep spindle detection is a difficult task and experts themselves often do not agree when annotating these offline. This disagreement is quantified by MODA [1] and represented in Table 4.1, row (1) for reference. The experts annotating the MODA dataset had an average performance of 0.72 on the whole cohort in term of the f1-score of their individual annotations w.r.t. the final labels. They are compared to other *offline detection*, i.e., when a virtually infinite computational budget and the whole signal is available, including future data points, presented under “offline detection” in Table 4.1 (taken from [1]). We instead perform *online detection*, which has additional challenges : (a) computation happens in real time ; (b) the future signal is not available.

ANN training and evaluation

The MODA dataset is relatively small (~ 24 h of annotated data) and heterogeneous. This adds some difficulty for training and properly assessing the performance of our models, because we choose to use only 10% of subjects as our validation set (for model selection), and another 10% of subjects as our test set (for final model evaluation). Since the overall results are dependent on the assignment of subjects to the three sets, we evaluate our models through the following procedure :

- we shuffle all subjects 10 times and compute a different training/validation/test split of the dataset each time ;
- for each split, we use the training set to train 3 models, the validation set being used to estimate their f1-score. We select the best of these 3 models by its best f1-score on the validation set. We then report the performance of this model in terms of its f1-score on the test set ;
- the above being repeated 10 times, we report the average test f1-score in Table 4.1, the corresponding standard deviation being indicated in parenthesis.

SpindleNet baseline

The SpindleNet [12] architecture is far too large to be implemented on the Portiloop, and it cannot run in real time unless using a high-end GPU/TPU Nevertheless, to the best of our knowledge, it is the state-of-the-art among previous work for the online detection of sleep spindles. Therefore, we use this architecture as a baseline for evaluating the performance of our own models, discovered via PMBO. Since SpindleNet is closed-source and trained on the MASS dataset, we retrain its architecture from scratch with the same pipeline as used to train our other classifiers. In particular, we balance training through oversampling (as opposed to the data augmentation technique used by the authors of the original paper), and more importantly we train and evaluate SpindleNet on the MODA dataset. The results of this experiment are presented in Table 4.1, row (2).

2-input model

We first derive a lightweight ANN architecture by drawing inspiration from SpindleNet. More precisely, we use PMBO to find a Pareto-optimal architecture that uses both the cleaned signal and the envelope as inputs. The resulting architecture is presented in Appendix 4.7.7⁸. Despite being small compared to the architectures commonly encountered in the literature,

⁸Appendices are available online at <https://mistlab.ca/papers/Portiloop/>

this architecture is still far too large to be synthesized in a fully-parallel fashion. Instead, the HLS compiler partially sequences operations in the FPGA. We measure a total duration of 40 ms for each forward pass in this model on the Portiloop. The detection performance of this model, reported in Table 4.1, row (3), outperforms the baseline. This is not particularly surprising, as the baseline was originally designed using MASS rather than MODA.

Envelope ablation

The idea of using the envelope along the raw signal as input to the ANN is drawn from the baseline. Since the envelope is computed from the raw signal, it should not contain any additional information that cannot be extracted by an ANN in theory ; we suspect it can be removed. To evaluate the relevance of this particular input, we perform the following ablation : to keep the same architecture (and thus the same model capacity), we replace one of the two inputs by a copy of the other. In Table 4.1, row (4) both inputs are the envelope, while in row (5) both inputs are the cleaned signal. We find that the envelope input can be removed : the model in which we replace the envelope with a copy of the cleaned signal (5) has the same performance as the original model (3), and performs marginally better on phase 2.

Single-input model

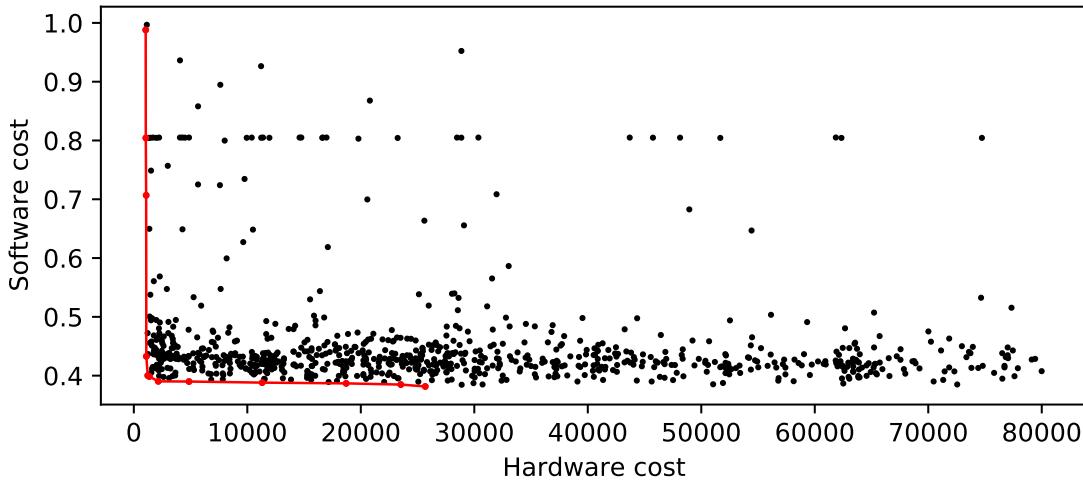


Figure 4.8: Single-input architecture search with PMBO. The hardware cost is the number of trainable parameters in the neural architecture, and the software cost is $1 - f_1$ -score of the fully-trained model. Black : non-Pareto-optimal models tested by the algorithm. Red dots : Pareto-optimal models found by the algorithm. Red line : Pareto front.

Since we deemed the use of the envelope input ineffective in the ablation study, we use PMBO one more time to devise our final Pareto-optimal ANN architecture, now with only the cleaned signal as input. For this matter, we run PMBO on 20 Tesla V100 GPU workers over a period of 24h. The detailed hyperparameters used in this experiment are provided in Appendix 4.7.3, and the results are visualized in Figure 4.8. PMBO brought the architecture search towards high-performance lightweight models. We select the best such model in terms of software cost *i.e.*, the model corresponding to the right-hand end of the Pareto front, since it is rather small with only 25.6k parameters. We measure the execution time of this architecture to be 20 ms per forward pass on the Portiloop (vs. 40 ms for the 2-input version).

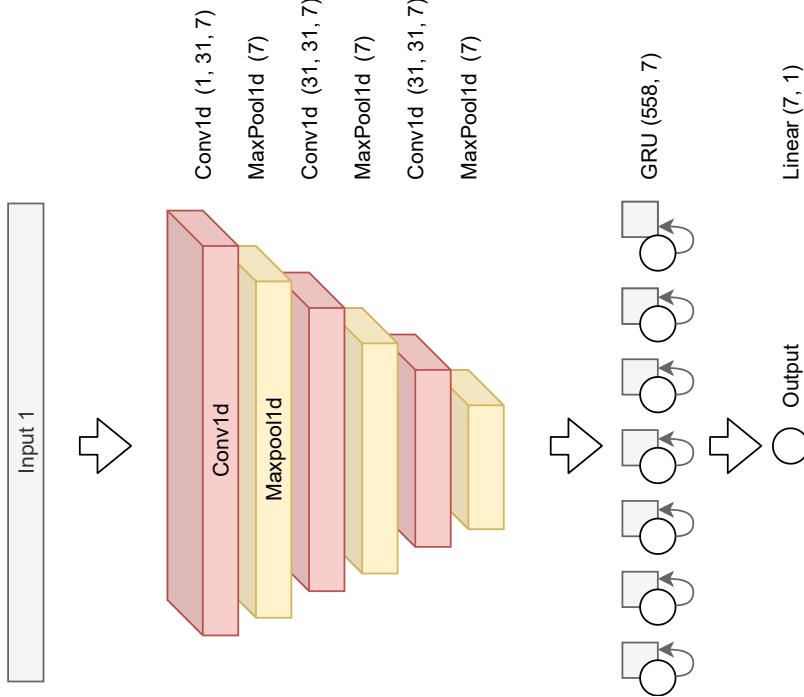


Figure 4.9: Final single-input ANN architecture. The dimensions of each layer are provided in parenthesis using the PyTorch nomenclature.

The selected architecture is described in Figure 4.9, and its detection performance is summarized in Table 4.1, row (6). Compared to our 2-input model, the single-input model exhibits the same performance, with even a marginal improvement on phase 2, while executing twice as fast. The detailed hyperparameters of this model are provided in Appendix 4.7.4.

Time-dilation ablation

We illustrate the importance of using virtual parallelization via time-dilation. This is done by shrinking the time-dilation (set at 168 ms by PMBO) to the minimum, *i.e.*, 20 ms since

this is the execution duration of the ANN per forward pass. This removes the virtual parallelization, since the same ANN must now be used for each sample. Therefore, each step of back-propagation reaches 8 times less far back in time during training. The result of this ablation is presented in Table 4.1, row (7). The highly deteriorated results illustrate the importance of time-dilation. This hints at the relevance of looking relatively far back in time to annotate sleep spindles.

Training and evaluating on different phases

We now perform an ablation on the MODA dataset itself to highlight a gap between the data of phase 1 (younger subjects) and the data of phase 2 (older subjects). Namely, we train the model on subjects drawn only from phase 1 on the one hand, and on subjects drawn only from phase 2 on the other hand. The results of these experiments are presented in Table 4.1, rows (8, 9). We observe that the ANN trained on phase 1 performs almost as well as the ANN trained on the whole cohort (6) on all subsets, including phase 2, whereas the ANN trained on phase 2 is noticeably worse on all subsets, even including phase 2. We hypothesize that this is because phase 2 has fewer sleep spindle examples, which are furthermore harder to detect. Using phase 2 during training is still useful in terms of generalization. Indeed, the ANN trained on phase 1 only (8) has a slightly worse performance when tested on phase 1 than the ANN trained on the whole cohort (6).

Regression

All models presented beforehand are classifiers. We also train a regressor with the same architecture, as explained in Section 4.3.6. There is a subtle difference in what this model measures when compared to our classifiers : whereas classifiers predict whether the signal is a sleep spindle according to the full definition given by MODA (including post-processing), the regressor predicts the mean score given by the experts (excluding post-processing). Since we are primarily interested in classification in this article, we find the threshold that maximizes the f1-score on the binary labels. This experiment is presented in Appendix 4.7.5. We find that the optimal threshold is 0.27 for phase 1, 0.23 for phase 2 and 0.26 for the whole cohort. We then evaluate the regressor with these thresholds on the classification task and report the results in Table 4.1, row (10). These results are slightly weaker than those of the classifier (6). We surmise that this effect comes from the post-processing steps performed by MODA to compute the binary labels. We choose the 1-input classifier (6) for the remainder of this article.

4.4.2 Real-time stimulation

The performance measured in Section 4.4.1 is not entirely representative of the performance on the final task. So far, we have only measured the capability of the model to annotate each data point of the signal individually. Yet, we want the ability to send one single stimulation per sleep spindle.

Qualitative results

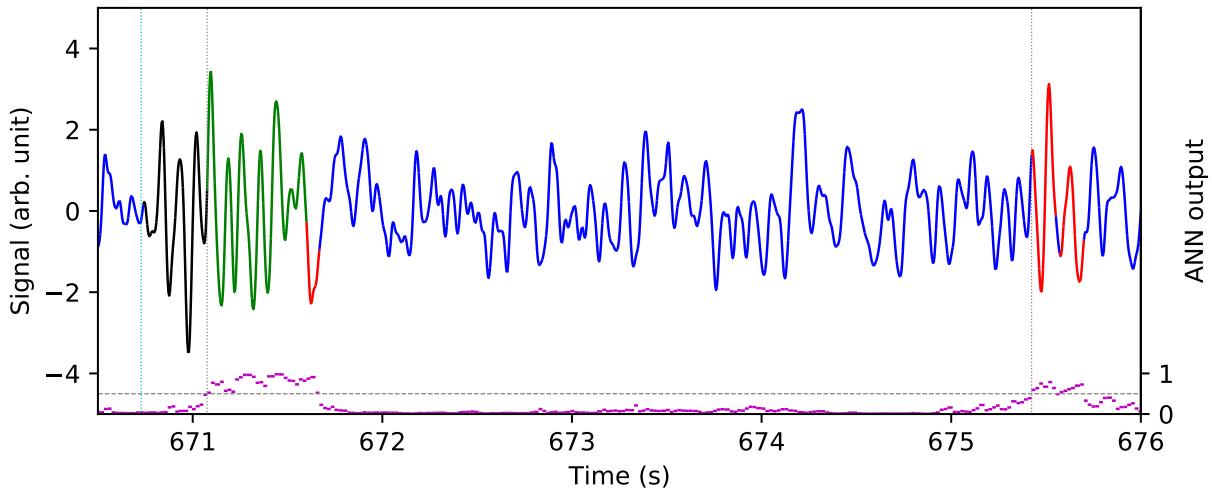


Figure 4.10: Stimulation with a 0.5 threshold. The color code is the same as Figure 4.2, with false positives being additionally displayed in red. Note the first part of the spindle is not detected (false negative), which introduces an ANN software delay. In addition, a small portion of the signal after the spindle is still detected as such (false positive), but it has no impact on our stimulation procedure. Finally, another portion of the signal that was not annotated as a spindle by MODA is detected as a spindle by our model (false positive), generating an undesirable stimulus is generated.

Figure 4.10 shows an example of imperfect sleep spindle detection and stimulation using our ANN (see Appendix 4.7.2 for more details on failure modes).

Although we chose using a classifier as opposed to a regressor, it is still possible to reduce incorrect stimulation by fine-tuning the detection threshold (0.5 by default). Figure 4.11 shows an increased threshold reduces the incorrect stimulation. However, a high threshold worsens the detection delay since it increases the number of false negatives at spindle onset.

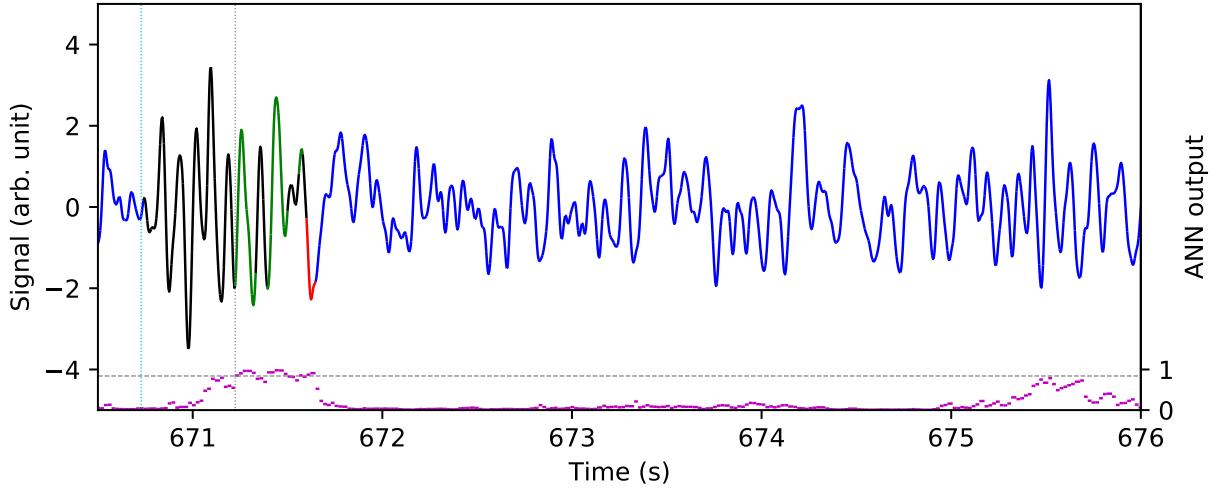


Figure 4.11: Stimulation with a 0.84 threshold.

Quantitative results

Although the results presented in Section 4.4.2 focus on the ANN detection delay, we must take in consideration the other sources of delay, *i.e.*, the software delay from FIRs (40 ms), the ANN forward pass duration (20 ms) and the stimulation hardware delay, to measure our real stimulation performance. The auditory stimulation delay in the Portiloop is 4 ms, for a total constant delay of 64 ms.

From now on, we redefine : (a) True positive : the first stimulus sent within the duration of a spindle, taking this additional delay into account ; (b) False positive : any other stimulus ; (c) False negative : any spindle that does not receive a stimulus within its labeled duration.

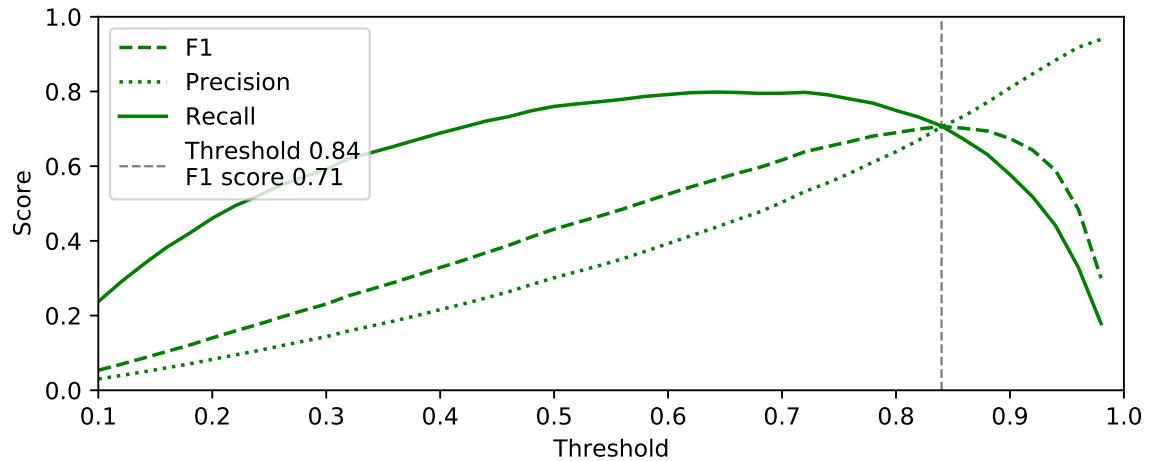


Figure 4.12: Evolution of the actual stimulation performance w.r.t. the chosen detection threshold.

Figure 4.12 displays the detection performance (taking all delays into account) of our final device. We compute the stimulation precision, recall and f1-score according to the aforementioned definitions of true positives, false positives and false negatives. This provides a visualization of possible trade-offs in terms of how many spindles we want to stimulate (recall) versus how sure we want to be that all stimuli are relevant (precision). In terms of f1-score, the best such trade-off is attained at a threshold of 0.84 with our model, giving a precision and a recall of 0.71.

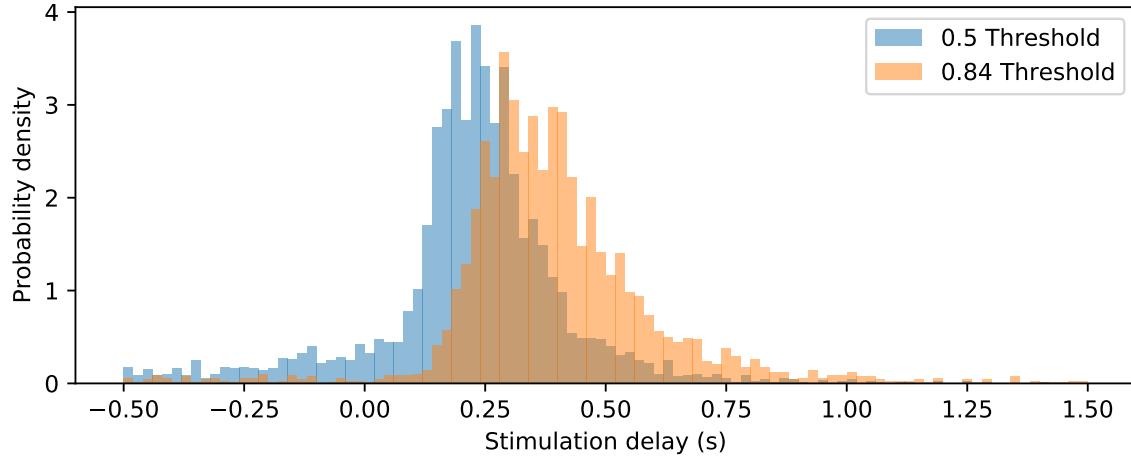


Figure 4.13: Distribution of stimulation delays for a classifier with 0.5 and 0.84 threshold. Delays are negative when spindles are stimulated in advance.

We further study the timing performance of our system. Figure 4.13 displays the distribution of stimulation delays, *i.e.*, the distribution of the stimulus being closest to the beginning of each sleep spindle, all delays being taken into account. Some of these delays are negative, as spindles are sometimes stimulated in advance (NB : we count these as false positives, which slightly harms our reported results). Moreover it shows the effect of increasing the detection threshold of our model on the stimulation delays. According to Figure 4.12, choosing a 0.84 detection threshold over the 0.5 default classification threshold in our ANN yields a better stimulation f1-score and in particular much more precise stimuli, but this comes at the price of slightly shifting the stimulation delay distribution to the right, *i.e.*, introducing some additional delay to the stimulation, as previously seen in Figure 4.11.

Finally, we estimate the Portiloop energy efficiency by running our final sleep spindle stimulation device continuously, powered by a fully-charged 20000 mAh battery. The battery dies out after 26 hours and 22 minutes, suggesting that our power consumption is roughly 756 mA.

4.5 Discussion and future work

The Portiloop system can be adapted to any application of EEG closed-loop stimulation. Although classifiers are most relevant in our case study, it would be straightforward to extend the Portiloop to quantitative tasks using regressors instead, as seen in Section 4.4.1.

As opposed to classical heuristics, our deep learning-based approach does not require specific knowledge of the phenomenon of interest when defining the classifier, nor requires a way to extract the relevant information. Instead, a large dataset of annotated signal suffices to derive a high-performance model that detects complex patterns such as sleep spindles. Once trained, techniques such as Integrated Gradients [56] can be used to better understand and fine-tune the ANN. Moreover, these techniques may help experts by revealing unknown dependencies in neural activity, as we hint in Appendix 4.7.1.

We have introduced PMBO to derive Pareto-optimal architectures in a parallel fashion. Although the algorithm produces high-performance lightweight architectures, we note that the predictions of the meta-learner are often near-constant in well-performing areas of the search space, suggesting that the meta-model could not further predict the software cost. We surmise that this is due to the large variance in model performance from one training session to another. This might be alleviated by training the same model several times, but we forbade this in our implementation to speed up the search.

Although we compare our architecture to the SpindleNet architecture, we did not have access to their weights and thus we could not compare their original model with ours on the MODA dataset. Instead, we retrained their architecture from scratch on MODA, using our own pipeline. Contrary to Kulkarni *et al.* [12], we could remove the envelope and power inputs without harming the performance of our models. During the course of our work, we have noticed that our 1-input model was less robust to shrinking the time-dilation than our 2-input model. Since the baseline does not use time-dilation, it cannot reach far back in time. We believe that this is important for the model to retrieve the information contained in these additional inputs. Note that it is necessary not to reduce the capacity of the model (*e.g.*, by cutting a branch of the ANN instead of duplicating inputs) when performing the ablation study, which Kulkarni *et al.* might have overlooked.

While the MODA dataset provides high-quality labels, training on a bigger dataset of similar quality would likely further improve the performance of our models. Expanding MODA is a relevant avenue for future work, as is implementing sim-to-real transfer, because the EEG acquisition and signal may differ somewhat from the training data. Transfer can be achieved with techniques such as domain randomization [57]. Alternatively, a dataset can be collected

on the target device and annotated following the same protocol as MODA.

Long term, we intend to target specific regions of sleep spindles for stimulation. This harder task will likely involve labeling these regions and developing more advanced RNNs/Transformers so as to consistently predict sleep spindles before they even start. Although our model does use information far back in time to make predictions (see Appendix 4.7.1), we believe that the main role currently played by the RNN is to accumulate information regarding whether the last few windows were spindles or not, rather than actually predicting the future (see Appendix 4.7.2). Such models will likely be more complex and computationally hungry. We will integrate a TPU in the system, as part of an ad-hoc PCB that will replace the prototyping board currently used, further miniaturizing the Portiloop. In general, finding an optimal model for a given Portiloop application involves either retraining our ANN, or re-executing PMBO with a new architecture. Both activities can be done by interested practitioners.

4.6 Conclusion

In this article, we introduce the Portiloop, a device that enables the real-time detection and stimulation of patterns of interest in EEG signal. Our system is open-source, portable, low-cost, and can be tailored for many such applications. The Portiloop enables implementing task-specific deep learning-based detectors directly in efficient hardware circuitry. This is made possible via an FPGA and a C++ HLS library that we open-source along with the rest of our code. We propose a pipeline to design neural architectures that are relevant for processing EEG signal in real time. We further propose PMBO, an algorithm that automates the process of finding efficient such models, using one to many parallel workers. We demonstrate our proposed system on the closed-loop stimulation of sleep spindles, a difficult task of high relevance for the neuroscience community. Our resulting system is the first portable device to be able to detect and stimulate sleep spindles in real time with a f1-score of 0.71, measured on MODA, a dataset renowned for the reliability of its labels. In the long range, the Portiloop will help the neuroscience community non-invasively explore brain functions, such as the role of sleep spindles in memory consolidation.

Acknowledgment

We thank Karine Lacourse for expert advice on spindle detection, and the MODA team for database access. Figures 4.1 and 4.3 use icons from <https://flaticon.com>.

4.7 Appendices

4.7.1 Model-based explanation of sleep spindles

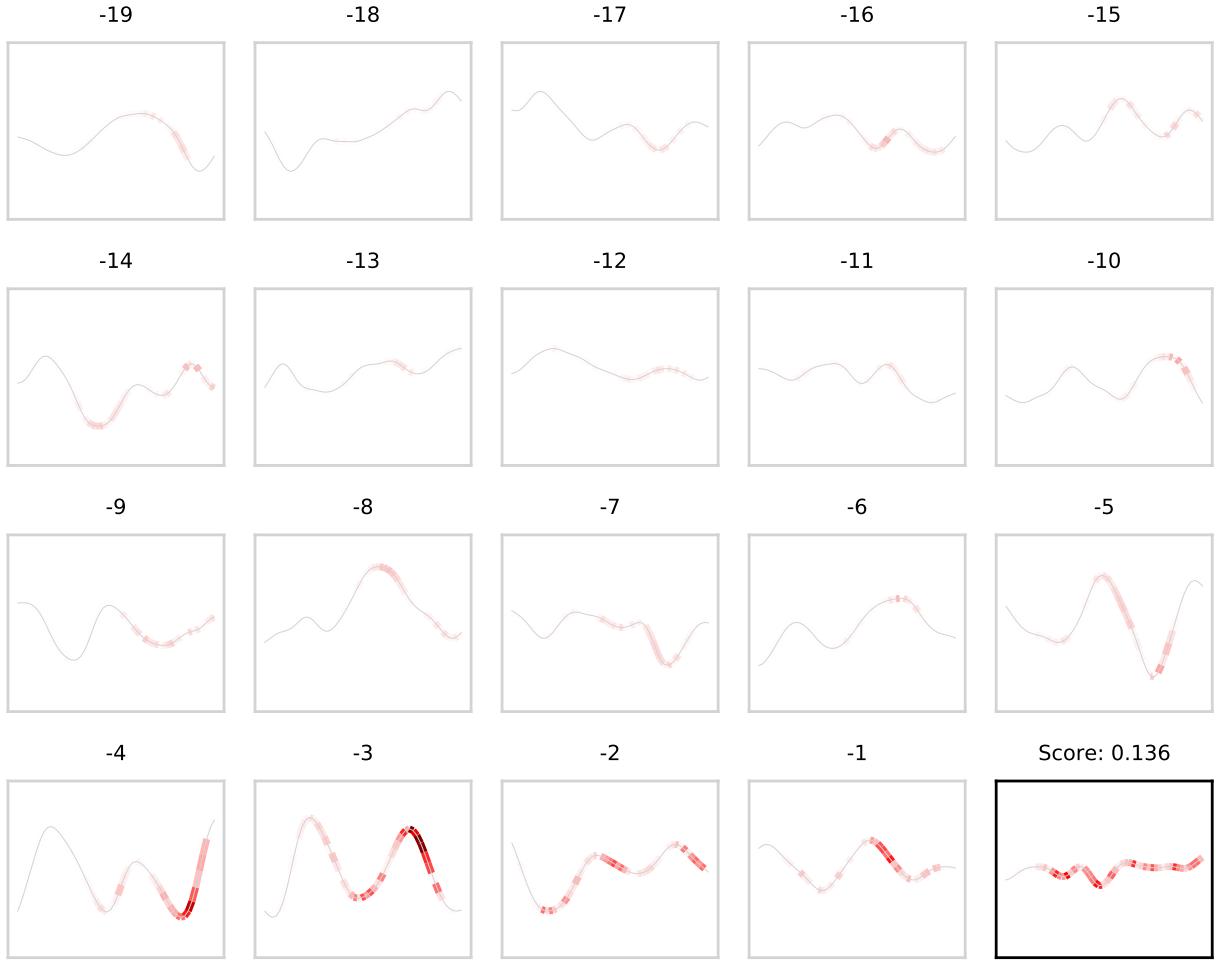


Figure 4.14: Integrated gradients (classifier ANN output : 0.136). The *integrated gradients* algorithm enables exploring why the model takes a given decision (the more a portion of the signal is represented in red, the highest its influence on the current output of the ANN). Grey windows are past inputs, whereas the black window is the current input : the past influences the current output due to the RNN. Here, the model finds that it is looking at the aftermath of a sleep spindle. With our time dilation and window size, a small portion of the window overlaps from one sample to the next. We see that this portion (at the left-hand side of each window) is in fact ignored by the model. Therefore, it is probably possible to shrink our model even more, although PMBO did not find this. In the future, this type of visualizations might also help experts better understand what sleep spindles are by revealing unknown influences.

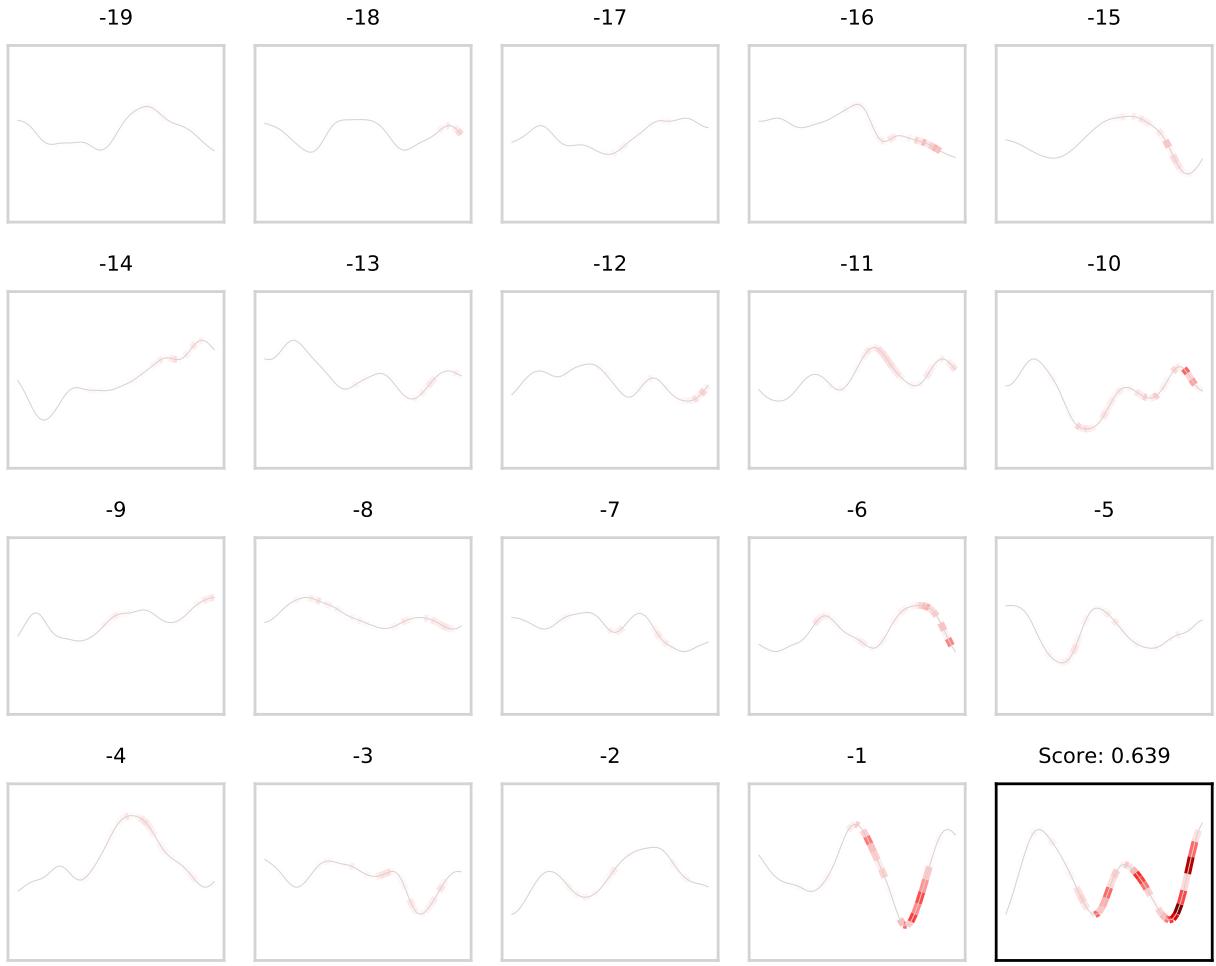


Figure 4.15: Integrated gradients (classifier ANN output : 0.639). The current window is within an actual sleep spindle. The model mainly focuses on the spindle itself, but also a few events that happened further back in time, to make its decision.

4.7.2 Stimulation visualizations

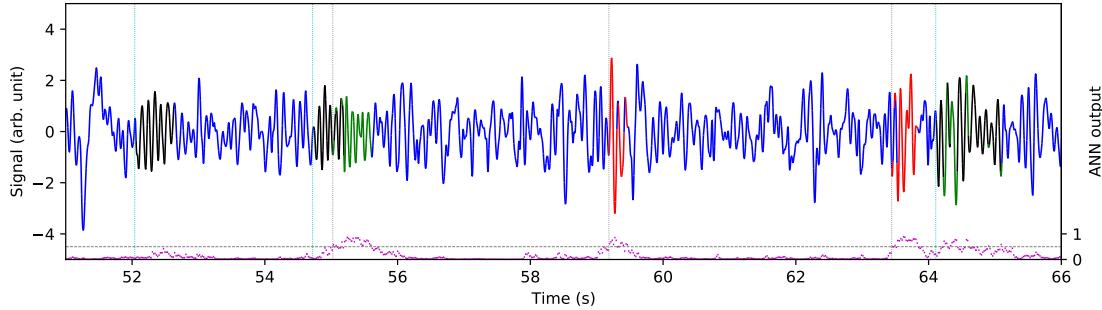


Figure 4.16: Different stimulation failure modes (classifier with threshold 0.5). Blue : no sleep spindle and no detection. Black : sleep spindle not detected. Green : sleep spindle correctly detected. Red : detection where the signal is not a spindle. Vertical blue : beginning of a spindle. Vertical grey : stimulation. Magenta : ANN output. Horizontal grey : detection threshold. This figure illustrates typical ‘failure’ cases of our final sleep spindle stimulating device. False negative : the first spindle is missed because the threshold is too big. True positive : the second spindle is correctly stimulated. False positive : a part of the signal not labeled as a sleep spindle by MODA is detected as a spindle by the device and stimulated. Almost true positive : the sleep spindle is stimulated in advance (NB : we count this case as a false positive).

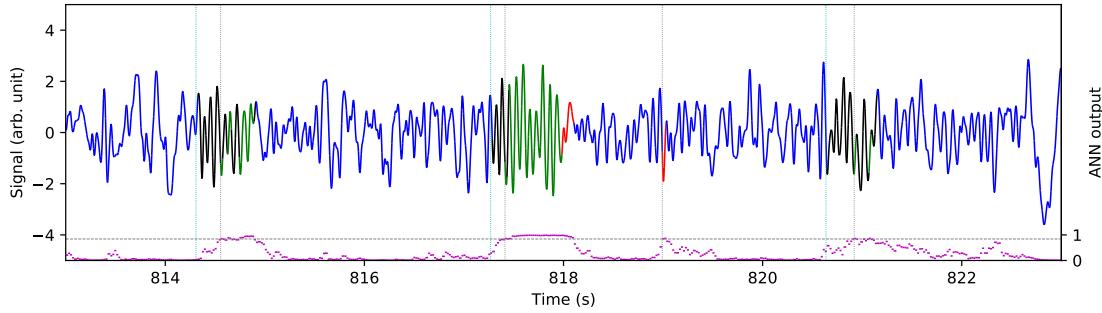


Figure 4.17: Classifier with threshold 0.84, success example. Same color code as Figure 4.16. Increasing the detection threshold from 0.5 to 0.84 removes most false positive stimuli (vertical grey not following vertical blue).

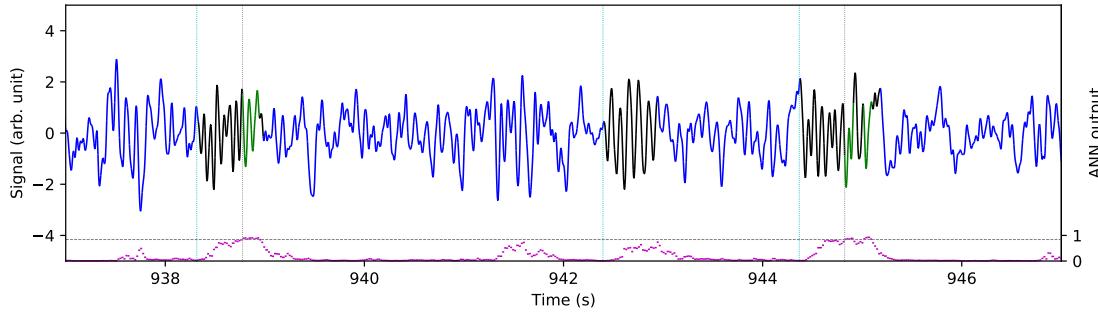


Figure 4.18: Classifier with threshold 0.84, failure example. Same color code as Figure 4.16. Increasing the detection threshold comes with more false negative stimuli (vertical blue not followed by vertical grey).

4.7.3 PMBO hyperparameters

Table 4.2: Hyperparameters used for PMBO

hyperparameter	selected value
range cost hardware	1000-80000
noise type 1	0,25
noise type 2	0,1
m	200
meta network type	MLP
# layers meta network	3
hidden size meta network	200
optimizer meta network	SGD
learning rate meta network	0.05
weight decay meta network	0.01

Two types of noise are used in our implementation of PMBO to foster exploration of the hyperparameter space :

- noise type 1 : portion of the m sampled models that are sampled randomly in the whole hyperparameter space, instead of in a Gaussian around the last completed experiment.
- noise type 2 : portion of the time when a model is sampled randomly in the m models, instead of being selected by its Pareto efficiency.

The ANN used for our meta model is a simple Multi-Layer Perceptron (MLP) of 3 fully connected layers. The hyperparameters we use in PMBO are summarized in Table 4.2.

4.7.4 Model hyperparameters

Table 4.3: Hyperparameters used to train the final model

Hyperparameter	Selected value	PMBO
Training		
optimizer	AdamW	
# epochs max	150	
epochs before early stopping	20	
early stopping running average factor	0,1	
batches per epoch	1000	
batch size	256	X
dropout on first layer	False	
dropout factor	0,5	
adam learning rate	0,0005	X
adam weight decay	0,01	
balancing mode	oversampling	
type of training	classification	
sequence length	50	
Architecture		
window size (s)	0,216	X
time dilation (s)	0,168	X
# CNN layers	3	X
# CNN channels	31	X
stride convolution	1	X
kernel size convolution	7	X
dilation convolution	1	X
stride max pooling	1	X
kernel size max pooling	1	X
dilation max pooling	1	X
RNN layers	1	X
RNN hidden size	7	X

The hyperparameters we use in our final model are listed in Table 4.3. Hyperparameters that were chosen by PMBO are marked with a cross under the PMBO column.

4.7.5 Best threshold for classifiers and regressors

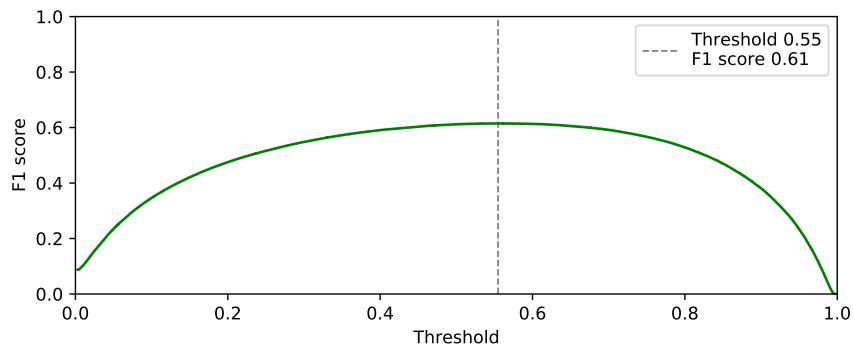


Figure 4.19: F1 score evolution with threshold variation on classification

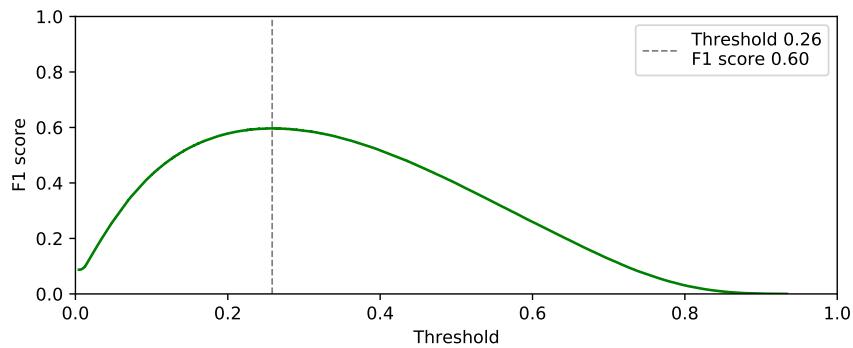


Figure 4.20: F1 score evolution with threshold variation on regression

4.7.6 Detection delay distributions

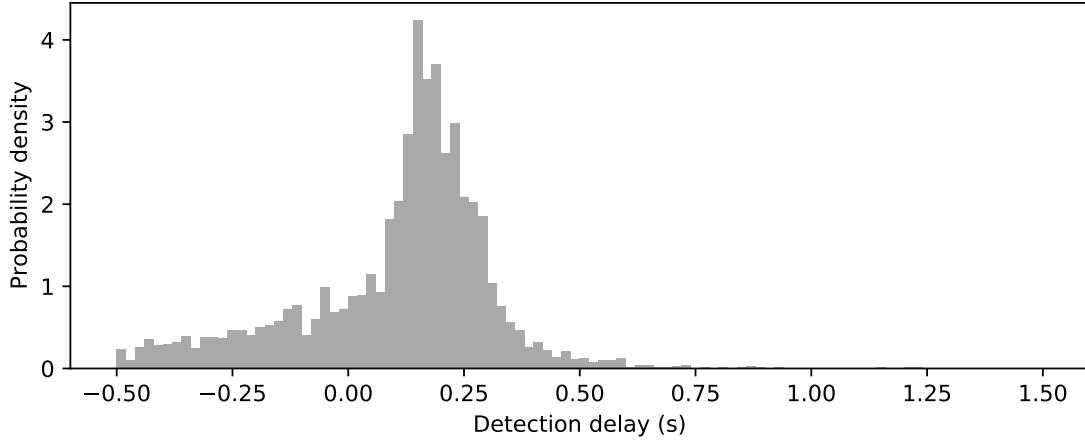


Figure 4.21: Stimulation histogram for 1 input network classification with a 0.25 threshold

4.7.7 2-input network architecture

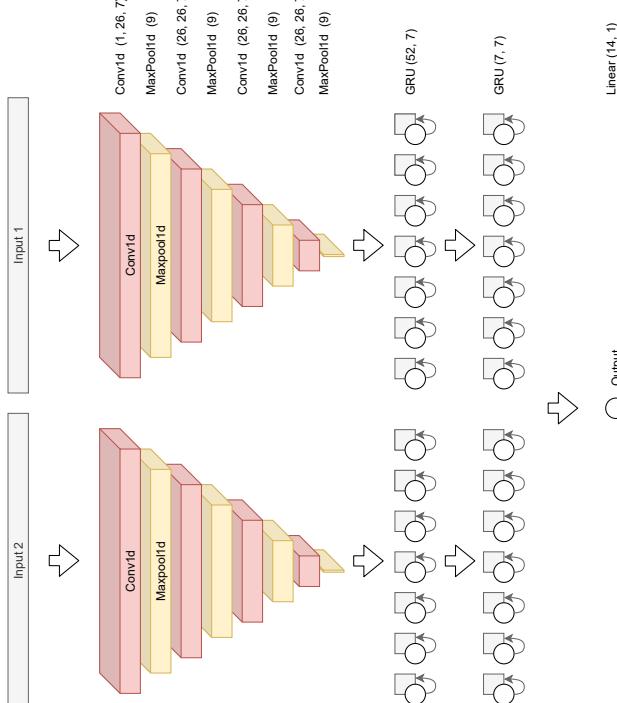


Figure 4.22: 2-input neural network architecture. This architecture consists of two sequences of CNN and RNN. The first sequence processes the cleaned raw signal (input 1), whereas the second processes the envelope (input 2). The latent features from both branches are concatenated and fed to a fully connected layer to yield the output of the ANN.

CHAPITRE 5 DISCUSSION GÉNÉRALE

L'objectif général de ce projet de maîtrise était de concevoir un appareil de stimulation en boucle fermée de l'activité cérébrale, et en particulier des « sleep spindles ». Ce système devait être portable, temps-réel, à bas coût et fonctionner sur batterie. Toutes ces contraintes ont été prises en compte pour le prototype présenté précédemment. Il permet, en plus de la stimulation des « sleep spindles », d'être adapté pour permettre d'autres applications. Même si les contraintes temps réel voulu à la création du projet n'ont pas encore été atteintes, ce dernier surpassé déjà l'état de l'art en la matière.

Le Portiloop permet d'acquérir et de traiter un signal EEG venant d'électrodes placées sur la tête d'un sujet, de transmettre ce signal à un réseau de neurones artificiels, entraîné à détecter un motif spécifique, qui détermine si un stimulus doit être envoyé ou non et de l'envoyer. Le tout en 64 ms avec une précision (pourcentage de stimuli correctement envoyés pendant la période du motif à détecter) et un rappel (pourcentage de motif ayant reçu un stimulus durant leur période) de 71%. Ces mesures n'ont, hélas, pas pu être réalisées sur les signaux acquis par l'appareil, car c'est une tâche qui demande l'acquisition et l'annotation d'une très grande quantité de données, qui sera faite par la suite, afin de confirmer les résultats déjà obtenus. L'acquisition a tout de même été testée sur moi-même pour valider la possibilité d'annoter visuellement les données. Il faudrait aller plus loin en utilisant simultanément le Portiloop avec un autre système d'acquisition déjà utilisé en neuroscience pour en comparer les résultats. Bien que l'étape d'acquisition et celle de détection aient été vérifiées séparément, il est tout à fait possible d'utiliser l'appareil pour faire de la stimulation en boucle fermée même si l'on n'est pas en mesure cependant d'assurer la qualité de cette stimulation.

Des améliorations sont déjà en cours sur l'appareil, comme sa conception en circuit imprimé (PCB) qui devrait permettre une miniaturisation et de meilleures performances (en améliorant le FPGA et le système audio).

Figures 5.1 et 5.2 présentent le rendu final du PCB avant sa mise en production. De plus, il existe une solution pour réaliser des réseaux de neurones artificiels embarqués et temps réel, autre que le FPGA, à savoir le Google Coral, qui contient un TPU. Il serait très intéressant d'essayer d'intégrer ce composant au reste du projet pour permettre de meilleures performances et sûrement de plus grosses architectures de réseaux.

Mais le projet ne s'arrête pas à la conception de l'appareil. En effet, afin d'optimiser la recherche du meilleur réseau de neurones qui fournirait de bons résultats tout en n'étant, ni trop lent, ni trop gros, pour être implémenté sur la carte, nous avons présenté un algorithme

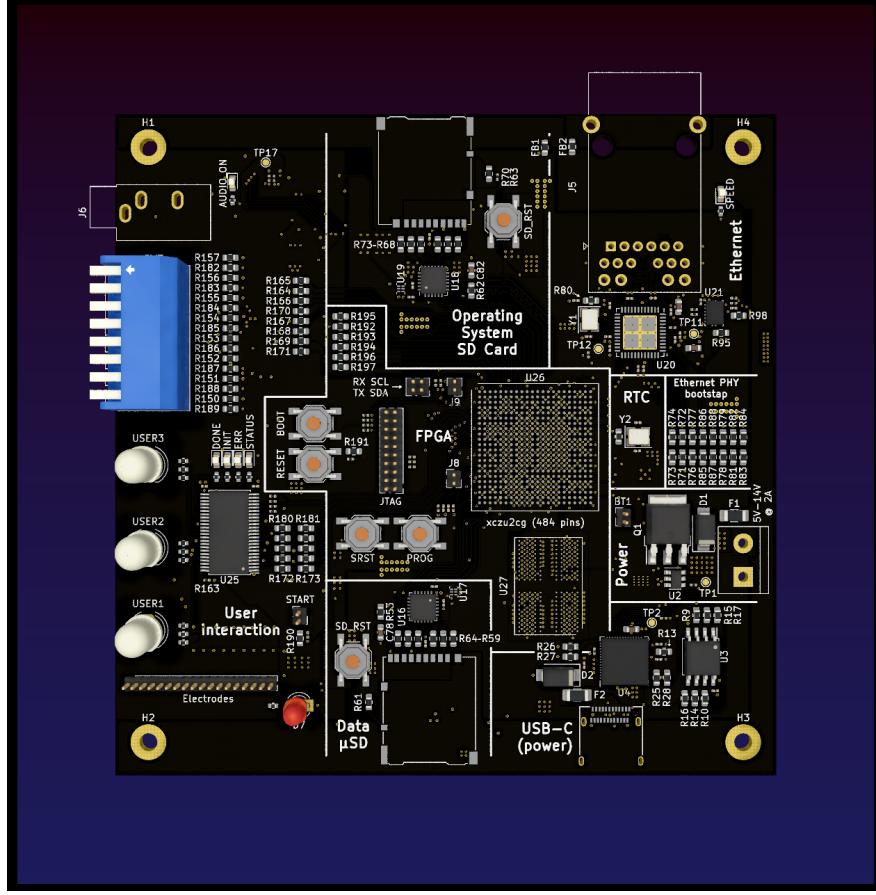


Figure 5.1: Face avant du PCB réalisé par © Xavier L'Heureux.

de recherche parallélisée d'hyperparamètres. Grâce à l'accès aux serveurs de Calcul Canada, il a été possible de trouver un réseau optimal en moins d'une journée, ce qui pouvait prendre des semaines lorsque la recherche était aléatoire. De plus, bien qu'il soit maintenant relativement accessible de développer des réseaux de neurones en Python, grâce notamment aux librairies Pytorch ou Tensorflow, il n'existe pas d'équivalents aussi répandus en FPGA. Il a donc fallu, en s'inspirant d'un projet qui a permis de fournir une base, mais qui n'a finalement pas été conservé, implémenter des réseaux de neurones en FPGA. Pour simplifier la tâche, le code n'a pas été fait en VHDL ou en Verilog, mais en C++ et il est compilé en langage FPGA par la suite. Ce code pourra être réutilisé pour d'autres projets cherchant à réaliser des réseaux similaires et pourra être adapté assez facilement. Cependant, il y a sûrement des optimisations à y faire, le logiciel utilisé (Vivado HLS) étant assez complexe et difficilement accessible aux novices du domaine.

Ce système, même dans sa version actuelle, va pouvoir être utilisé pour mener des études cliniques afin d'en apprendre plus sur les « sleep spindles » et leur rôle dans la consolidation

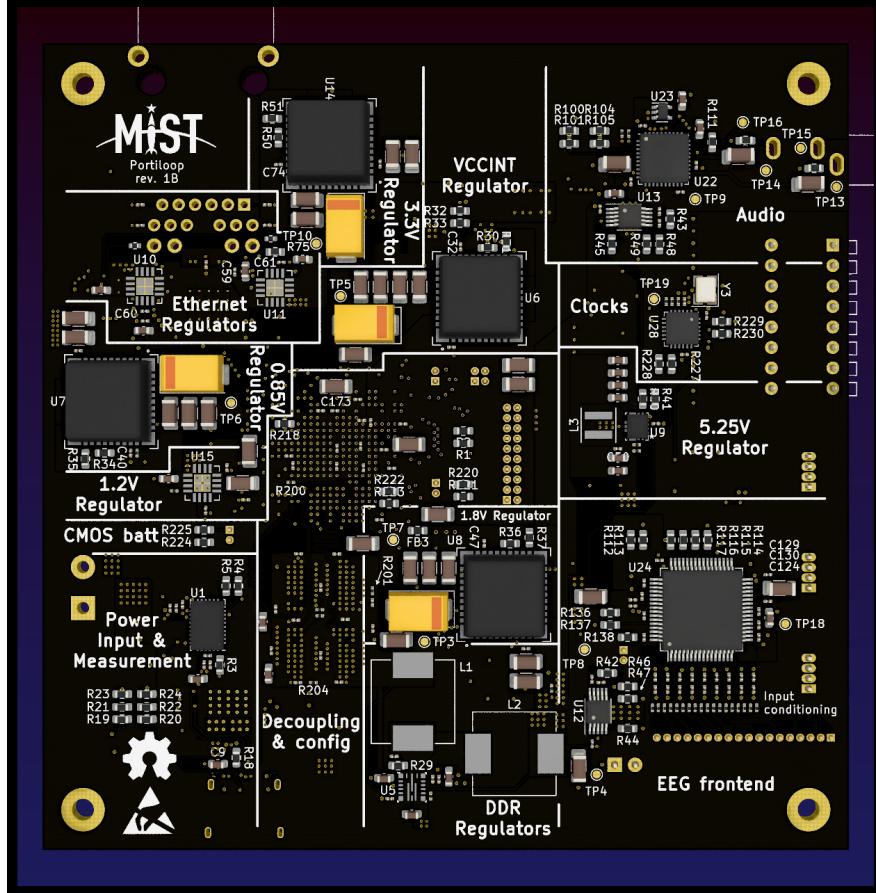


Figure 5.2: Face arrière du PCB réalisé par © Xavier L’Heureux.

de la mémoire. Le laboratoire CLASP d’Emily Coffey à l’Université de Concordia, avec qui ce projet a été réalisé, a déjà pour but de mener ces études, mais d’autres laboratoires ont déjà fait part de leur intérêt pour le projet, dû au manque d’alternative dans le sujet. De plus, le Portiloop pouvant s’adapter à d’autres applications, il est possible et souhaitable que son utilisation aille plus loin que l’étude de cas des « sleep spindles ». Lorsque des thérapies auront été trouvées, il sera possible de commercialiser le produit, rappelant une fois de plus l’importance de sa portabilité et de son bas coût.

CHAPITRE 6 CONCLUSION

Le projet de recherche présenté dans ce mémoire a cherché à concevoir un appareil de stimulation en boucle fermée de l'activité cérébrale, à la fois portable et à bas coût. Il a été testé sur une étude de cas : la stimulation de « sleep spindles », un type d'oscillations de l'activité cérébrale pendant le sommeil, qui joue un rôle dans la consolidation de la mémoire.

6.1 Synthèse des Travaux

La finalité de tous les travaux se retrouve dans le prototype présenté chapitre 4. Le système fait l'acquisition et le traitement du signal EEG. Il est capable d'utiliser un réseau de neurones artificiels implémenté en FPGA pour détecter des motifs spécifiques. Un stimulus audio ou une impulsion déclenchant un système de stimulation sont aussi disponibles. Il s'exécute en temps réel, et dans l'étude de cas des « sleep spindles », réussit en ~ 300 ms à les stimuler avec une précision et un rappel de 71%, en utilisant un réseau de neurones développé pour le projet. Celui-ci a été réalisé à l'aide d'un algorithme de recherche parallélisée d'hyperparamètres et d'une librairie permettant de créer des réseaux facilement en FPGA. Tous les deux ont été introduits dans ce mémoire. L'appareil est embarqué et il fonctionne sur batterie pendant au moins 26 heures. Son coût est un inférieur à 500 \$ CA.

6.2 Limitations

Cependant, le signal acquis et traité par le système n'a pu être validé que partiellement. Seule une validation visuelle a été réalisée, sur un seul sujet. Ce qui n'est pas suffisant pour attester de sa qualité. Le jeu de données utilisé pour entraîner et tester le réseau de neurones a semblé trop petit, et il n'est pas sûr qu'un réseau entraîné sur ces données fonctionne correctement sur les données du Portiloop. Même si l'étude de cas permet de présenter une possible utilisation du projet, et met en avant son adaptabilité, il n'est cependant pas trivial de créer une nouvelle application.

6.3 Améliorations Futures

La réalisation d'un circuit intégré pourra permettre au Portiloop une miniaturisation et une utilisation plus adaptées de ses composants. Ainsi certains composants très sollicités, tels que le FPGA, pourront être améliorés et beaucoup de fonctionnalités non nécessaires de la

Pynq Z2 pourront être supprimées. Il pourra être judicieux de tester le Google Coral pour y réaliser l'inférence du modèle. L'acquisition et l'annotation de données avec l'appareil, venant en complément du jeu de données actuel, permettront de s'assurer et d'améliorer les performances de détection. Pour aider d'autres chercheurs à utiliser l'appareil, il serait intéressant de présenter une autre étude de cas, pour généraliser une fois de plus ce que l'appareil est capable de faire.

RÉFÉRENCES

- [1] K. Lacourse, B. Yetton, S. Mednick et S. C. Warby, “Massive online data annotation, crowdsourcing to generate high quality sleep spindle annotations from EEG data,” *Sci Data*, vol. 7, n°. 1, p. 190, juin 2020.
- [2] R. Zelmann, A. C. Paulk, I. Basu, A. Sarma, A. Yousefi, B. Crocker, E. Eskandar, Z. Williams, G. R. Cosgrove, D. S. Weisholtz, D. D. Dougherty, W. Truccolo, A. S. Widge et S. S. Cash, “CLoSES : A platform for closed-loop intracranial stimulation in humans,” *NeuroImage*, vol. 223, p. 117314, déc. 2020.
- [3] U. Ha et H.-J. Yoo, “A multimodal drowsiness monitoring ear-module system with closed-loop real-time alarm,” dans *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, oct. 2016, p. 536–539.
- [4] C.-T. Lin, Y.-C. Chen, T.-Y. Huang, T.-T. Chiu, L.-W. Ko \$*\$, S.-F. Liang, H.-Y. Hsieh, S.-H. Hsu et J.-R. Duann, “Development of Wireless Brain Computer Interface With Embedded Multitask Scheduling and its Application on Real-Time Driver’s Drowsiness Detection and Warning,” *IEEE Transactions on Biomedical Engineering*, vol. 55, n°. 5, p. 1582–1591, mai 2008.
- [5] A. von Lühmann, J. Addesa, S. Chandra, A. Das, M. Hayashibe et A. Dutta, “Neural interfacing non-invasive brain stimulation with NIRS-EEG joint imaging for closed-loop control of neuroenergetics in ischemic stroke,” dans *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*, mai 2017, p. 349–353.
- [6] N. Kosmyna et P. Maes, “Attentivu : An EEG-based closed-loop biofeedback system for real-time monitoring and improvement of engagement for personalized learning,” *Sensors*, vol. 19, n°. 23, 2019.
- [7] C. M. McCrimmon, J. L. Fu, M. Wang, L. S. Lopes, P. T. Wang, A. Karimi-Bidhendi, C. Y. Liu, P. Heydari, Z. Nenadic et A. H. Do, “Performance Assessment of a Custom, Portable, and Low-Cost Brain-Computer Interface Platform,” *IEEE Transactions on Biomedical Engineering*, vol. 64, n°. 10, p. 2313–2320, 2017.
- [8] S. Zotou, G. K. Kostopoulos et T. A. Antonakopoulos, “Real-time Spindles Detection for Acoustic Neurofeedback,” dans *Brain Function Assessment in Learning*, ser. Lecture Notes in Computer Science, C. Frasson et G. Kostopoulos, édit. Cham : Springer International Publishing, 2017, p. 159–168.

- [9] H.-V. V. Ngo, T. Martinetz, J. Born et M. Mölle, "Auditory Closed-Loop Stimulation of the Sleep Slow Oscillation Enhances Memory," *Neuron*, vol. 78, n°. 3, p. 545–553, mai 2013.
- [10] H.-V. V. Ngo, M. Seibold, D. C. Boche, M. Mölle et J. Born, "Insights on auditory closed-loop stimulation targeting sleep spindles in slow oscillation up-states," *Journal of Neuroscience Methods*, vol. 316, p. 117–124, mars 2019.
- [11] L. L. Chen, R. Madhavan, B. I. Rapoport et W. S. Anderson, "Real-time brain oscillation detection and phase-locked stimulation using autoregressive spectral estimation and time-series forward prediction," *IEEE Trans. on Biomed. Eng.*, vol. 60, n°. 3, p. 753–762, 2013.
- [12] P. M. Kulkarni, Z. Xiao, E. J. Robinson, A. S. Jami, J. Zhang, H. Zhou, S. E. Henin, A. A. Liu, R. S. Osorio, J. Wang et Z. Chen, "A deep learning approach for real-time detection of sleep spindles," *J. Neural Eng.*, vol. 16, n°. 3, p. 036004, juin 2019.
- [13] N. I. Tapia et P. A. Estévez, "RED : Deep Recurrent Neural Networks for Sleep EEG Event Detection," *arXiv :2005.07795*, mai 2020.
- [14] S. Chambon, V. Thorey, P. J. Arnal, E. Mignot et A. Gramfort, "A deep learning architecture to detect events in EEG signals during sleep," dans *28th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2018, September 17, 2018 - September 20, 2018*, ser. IEEE International Workshop on Machine Learning for Signal Processing, MLSP, vol. 2018-September. Aalborg, Denmark : IEEE Computer Society, 2018, p. IEEE Signal Processing Society.
- [15] C. R. Patti, S. S. Shahrbabaki, C. Dissanayaka et D. Cvetkovic, "Application of random forest classifier for automatic sleep spindle detection," dans *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, oct. 2015, p. 1–4.
- [16] C. O'Reilly, N. Gosselin, J. Carrier et T. Nielsen, "Montreal Archive of Sleep Studies : An open-access resource for instrument benchmarking and exploratory research," *Journal of Sleep Research*, vol. 23, n°. 6, p. 628–635, 2014.
- [17] K. Lacourse, J. Delfrate, J. Beaudry, P. Peppard et S. C. Warby, "A sleep spindle detection algorithm that emulates human expert spindle scoring," *J. Neuroscience Methods*, vol. 316, p. 3–11, mars 2019.
- [18] F. Ferrarelli, R. Huber, M. J. Peterson, M. Massimini, M. Murphy, B. A. Riedner, A. Watson, P. Bria et G. Tononi, "Reduced Sleep Spindle Activity in Schizophrenia Patients," *AJP*, vol. 164, n°. 3, p. 483–492, mars 2007.

- [19] M. Mölle, L. Marshall, S. Gais et J. Born, “Grouping of Spindle Activity during Slow Oscillations in Human Non-Rapid Eye Movement Sleep,” *J. Neurosci.*, vol. 22, n°. 24, p. 10 941–10 947, déc. 2002.
- [20] N. Martin, M. Lafourte, J. Godbout, M. Barakat, R. Robillard, G. Poirier, C. Bastien et J. Carrier, “Topography of age-related changes in sleep spindles,” *Neurobiology of Aging*, vol. 34, n°. 2, p. 468–476, févr. 2013.
- [21] E. J. Wamsley, M. A. Tucker, A. K. Shinn, K. E. Ono, S. K. McKinley, A. V. Ely, D. C. Goff, R. Stickgold et D. S. Manoach, “Reduced Sleep Spindles and Spindle Coherence in Schizophrenia : Mechanisms of Impaired Memory Consolidation ?” *Biological Psychiatry*, vol. 71, n°. 2, p. 154–161, janv. 2012.
- [22] L. Ray, S. Sockeel, M. Soon, A. Bore, A. Myhr, B. Stojanoski, R. Cusack, A. M. Owen, J. Doyon et S. Fogel, “Expert and crowd-sourced validation of an individualized sleep spindle detection method employing complex demodulation and individualized normalization,” *Front. Hum. Neurosci.*, vol. 9, 2015.
- [23] A. Parekh, I. W. Selesnick, D. M. Rapoport et I. Ayappa, “Detection of K-complexes and sleep spindles (DETOKS) using sparse optimization,” *Journal of Neuroscience Methods*, vol. 251, p. 37–46, août 2015.
- [24] S.-F. Liang, C.-E. Kuo, Y.-H. Hu, C.-Y. Chen et Y.-H. Li, “An adaptive neuro-fuzzy inference system for sleep spindle detection,” dans *2012 International Conference on Fuzzy Theory and Its Applications (iFUZZY2012)*, nov. 2012, p. 369–373.
- [25] N. Yasuhara, T. Natori, M. Hayashi et N. Aikawa, “A Study on Automatic Detection of Sleep Spindles using a Long Short-Term Memory Network,” dans *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, août 2019, p. 45–48.
- [26] D. Tan, R. Zhao, J. Sun et W. Qin, “Sleep spindle detection using deep learning : A validation study based on crowdsourcing,” dans *37th IEEE Engineering in Medicine and Biology Conference (EMBC)*, août 2015, p. 2828–2831.
- [27] M. Vohra et S. Fasciani, “PYNQ-Torch : A framework to develop PyTorch accelerators on the PYNQ platform,” dans *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, déc. 2019, p. 1–6.
- [28] Y. Hao et S. Quigley, “The implementation of a deep recurrent neural network language model on a xilinx fpga,” *arXiv preprint arXiv :1710.10296*, 2017.
- [29] Z. Yin, W. Gross et B. H. Meyer, “Probabilistic sequential multi-objective optimization of convolutional neural networks,” dans *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, p. 1055–1060.

- [30] G. Zarubin, C. Gundlach, V. Nikulin et M. Bogdan, “Real-time phase detection for EEG-based tACS closed-loop system,” dans *6th International Congress on Neurotechnology, Electronics and Informatics*, Seville, Spain, 2018, p. 13–20.
- [31] S. Shirinpour, I. Alekseichuk, K. Mantell et A. Opitz, “Experimental evaluation of methods for real-time EEG phase-specific transcranial magnetic stimulation,” *J. Neural Engineering*, vol. 17, n°. 4, 2020.
- [32] C. Zrenner, P. Belardinelli, F. Müller-Dahlhaus et U. Ziemann, “Closed-loop neuroscience and non-invasive brain stimulation : a tale of two loops,” *Frontiers in cellular neuroscience*, vol. 10, p. 92, 2016.
- [33] A. Vassileva, D. van Blooij, F. Leijten et G. Huiskamp, “Neocortical electrical stimulation for epilepsy : Closed-loop versus open-loop,” *Epilepsy research*, vol. 141, p. 95–101, 2018.
- [34] J. Choi, M. Kwon et S. C. Jun, “A systematic review of closed-loop feedback techniques in sleep studies—related issues and future directions,” *Sensors*, vol. 20, n°. 10, p. 2770, 2020.
- [35] K. D. Fehér, M. Wunderlin, J. G. Maier, E. Hertenstein, C. Schneider, C. Mikutta, M. A. Züst, S. Klöppel et C. Nissen, “Shaping the slow waves of sleep : A systematic and integrative review of sleep slow wave modulation in humans using non-invasive brain stimulation,” *Sleep medicine reviews*, p. 101438, 2021.
- [36] F. Salfi, A. D’Atri, D. Tempesta, L. De Gennaro et M. Ferrara, “Boosting slow oscillations during sleep to improve memory function in elderly people : A review of the literature,” *Brain Sciences*, vol. 10, n°. 5, p. 300, 2020.
- [37] M. O. Harrington et S. A. Cairney, “Sounding it out : auditory stimulation and overnight memory processing,” *Current Sleep Medicine Reports*, 2021.
- [38] T. O. Bergmann et B. P. Staresina, “Neuronal oscillations and reactivation subserving memory consolidation,” dans *Cognitive neuroscience of memory consolidation*. Springer, 2017, p. 185–207.
- [39] B. Rasch et J. Born, “About sleep’s role in memory,” *Physiological reviews*, 2013.
- [40] S. M. Fogel et C. T. Smith, “Learning-dependent changes in sleep spindles and stage 2 sleep,” *Journal of sleep research*, vol. 15, n°. 3, p. 250–255, 2006.
- [41] M. Lafontaine, J.-F. Gagnon, N. Martin, V. Latreille, J. Dubé, M. Bouchard, C. Bastien et J. Carrier, “Sleep spindles and rapid eye movement sleep as predictors of next morning cognitive performance in healthy middle-aged and older participants,” *Journal of sleep research*, vol. 23, n°. 2, p. 159–167, 2014.

- [42] S. Fogel, C. Vien, A. Karni, H. Benali, J. Carrier et J. Doyon, “Sleep spindles : a physiological marker of age-related changes in gray matter in brain regions supporting motor skill memory consolidation,” *Neurobiology of aging*, vol. 49, p. 154–164, 2017.
- [43] L. M. J. Fernandez et A. Lüthi, “Sleep Spindles : Mechanisms and Functions,” *Physiological Reviews*, vol. 100, n°. 2, p. 805–868, 2020.
- [44] K. R. Peters, L. B. Ray, S. Fogel, V. Smith et C. T. Smith, “Age differences in the variability and distribution of sleep spindle and rapid eye movement densities,” *PLoS one*, vol. 9, n°. 3, p. e91047, 2014.
- [45] S. Purcell, D. Manoach, C. Demanuele, B. Cade, S. Mariani, R. Cox, G. Panagiotaropoulou, R. Saxena, J. Pan, J. Smoller *et al.*, “Characterizing sleep spindles in 11,630 individuals from the national sleep research resource,” *Nature communications*, vol. 8, n°. 1, p. 1–16, 2017.
- [46] P. Albouy, A. Weiss, S. Baillet et R. J. Zatorre, “Selective entrainment of theta oscillations in the dorsal stream causally enhances auditory working memory performance,” *Neuron*, vol. 94, n°. 1, p. 193–206, 2017.
- [47] R. Xu, N. Jiang, C. Lin, N. Mrachacz-Kersting, K. Dremstrup et D. Farina, “Enhanced low-latency detection of motor intention from EEG for closed-loop brain-computer interface applications,” *IEEE Transactions on Biomedical Engineering*, vol. 61, n°. 2, p. 288–296, 2014.
- [48] *Pynq Z2 Reference Manual*, TUL, 2019, rev. 1.1.
- [49] *ADS1299-x Low-Noise, 4-, 6-, 8-Channel, 24-Bit, Analog-to-Digital Converter for EEG and Biopotential Measurements datasheet*, Texas Instrument, 2012, rev. C (2017).
- [50] J. Chung, C. Gulcehre, K. Cho et Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” *arXiv :1412.3555 [cs]*, déc. 2014.
- [51] S. Yang, X. Yu et Y. Zhou, “LSTM and GRU Neural Network Performance Comparison Study : Taking Yelp Review Dataset as an Example,” dans *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, juin 2020, p. 98–101.
- [52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser et I. Polosukhin, “Attention is all you need,” dans *Advances in neural information processing systems*, 2017, p. 5998–6008.
- [53] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. Hasegawa-Johnson et T. S. Huang, “Dilated recurrent neural networks,” *arXiv preprint arXiv :1710.02224*, 2017.

- [54] Y. Yang, K. Zha, Y.-C. Chen, H. Wang et D. Katabi, “Delving into deep imbalanced regression,” *arXiv preprint arXiv :2102.09554*, 2021.
- [55] C. Iber, S. Ancoli-Israel, A. L. Chesson, S. F. Quan *et al.*, *The AASM manual for the scoring of sleep and associated events : rules, terminology and technical specifications*. American academy of sleep medicine Westchester, IL, 2007, vol. 1.
- [56] M. Sundararajan, A. Taly et Q. Yan, “Axiomatic attribution for deep networks,” dans *International Conference on Machine Learning*. PMLR, 2017, p. 3319–3328.
- [57] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba et P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” dans *International conference on intelligent robots and systems (IROS)*. IEEE, 2017, p. 23–30.