University of Nottingham
School of Computer Science

A benchmarking framework for improving machine learning with fNIRS neuroimaging data

Thesis
PhD in Computer Science

Johann Benerradi

Supervisors
Dr Max L. Wilson
Prof Michel F. Valstar
Dr Jeremie Clos

Examiners
Dr Paola Pinti
Dr Jamie Twycross

November 2023

Abstract

Functional near-infrared spectroscopy (fNIRS) is a non-invasive neuroimaging technology that presents attractive advantages such as a good compromise between temporal resolution, spatial resolution, and portability. Thanks to its properties and relative robustness to motion, it is often used in naturalistic settings and one of its key domains of application is the study of working memory and mental workload. However fNIRS is less popular and tested than other technologies such as electroencephalography (EEG) or functional magnetic resonance imaging (fMRI) which can nowadays be used in a clinical context. As such, there is no real consensus yet in the research community as to how fNIRS data should be used and analysed. While efforts to establish best practices with fNIRS have been published, there are still no community standards for using machine learning with fNIRS data. Moreover, the lack of open source benchmarks and standard expectations for reporting means that published works often claim high generalisation capabilities, but with poor methodology practices or reporting with missing details. These issues make it hard to evaluate the performance of models when it comes to choosing them for brain-computer interface (BCI) applications.

In this thesis, we present the creation of an open-source benchmarking framework called *BenchNIRS* to establish a best practice machine learning methodology to develop, evaluate, and compare models for classification from fNIRS data, using open-access datasets from the literature. This framework makes the implementation of a robust machine learning methodology for fNIRS much simpler and less time-consuming.

We demonstrate the utility of the framework by presenting a benchmarking of 6 baseline machine learning models (linear discriminant analysis (LDA), support vector machine (SVM), k-nearest neighbours (kNN), artificial neural network (ANN), convolutional neural network (CNN) and long short-term memory (LSTM)) on 5 open-access datasets and investigate the influence of different factors on the classification performance. Those benchmarks set a solid basis for future comparisons of machine learning approaches for fNIRS classification and reveal that most models have lower performances than expected when evaluating them in an unbiased way.

We then go on to use the framework in a specific challenging use case, the classification of mental workload, and demonstrate how it is used to develop machine learning models tailored to a specific task or dataset. We show that models using as inputs longer durations of fNIRS recordings did not necessarily predict n-back levels better and that the classification from fNIRS data on this task is relatively challenging, despite the tailoring of deep learning models to specific device configurations being promising.

Finally, we go beyond supervised learning and extend the framework to study how unlabelled segments of the data can be used to improve classification. This leads us to perform transfer learning with a self-supervised representation learning pretext task and study to what extent it can be useful to fNIRS data classification. We explore the use of opposite hemoglobin type reconstruction as a pretext task and extend the framework to support the exploration of more pretext tasks for transfer learning in the future.

Thank you…

To my examiners, who so benevolently challenged me.

To my supervisors, who so kindly guided me.

To my mentors, who so greatly inspired me.

To my dear colleagues, who so pedagogically helped me.

To my friends, who so generously supported me.

To my family, who so unconditionally loved me.

Contents

Acronyms

AI artificial intelligence. 92

ANN artificial neural network. 1, 11, 27, 32, 44, 45, 51–60, 62, 65, 67, 69, 70, 76, 98, 99

API application programming interface. 37, 42, 43, 54, 99

BCI brain-computer interface. 1, 7, 9–13, 25, 26, 31–33, 35, 36, 61, 62, 64, 66, 78, 88, 90–93, 96, 97, 99

BOLD blood oxygenation level-dependent. 9, 16

CBSI correlation based signal improvement. 18

CED convolutional encoder-decoder. 79–81

CNN convolutional neural network. 1, 11, 25, 29, 32, 44–46, 51–53, 56, 57, 61–63, 65, 67–76, 78, 84, 85, 91, 98–100

CSP common spatial pattern. 25

DPF differential pathlength factor. 39, 52

EDA electrodermal activity. 23, 25

EEG electroencephalography. 1, 7, 16, 23, 25, 26, 35, 78

fMRI functional magnetic resonance imaging. 1, 7, 9, 16, 23

fNIRS functional near-infrared spectroscopy. 1, 7–17, 19, 23, 25, 26, 31–33, 35–37, 39, 40, 42, 43, 47–53, 55, 62, 64–66, 68, 71, 72, 76–80, 85, 87–101

HbO oxyhaemoglobin. 13–16, 18, 19, 38–40, 52, 55, 56, 59, 60, 63, 67, 71, 72, 74, 78, 79, 81, 85, 86, 92, 100

HbR deoxyhaemoglobin. 13–16, 18, 19, 23, 38–40, 52, 55, 56, 59, 60, 63, 67, 71, 72, 74, 78, 79, 81, 85, 86, 92, 100

HCI human-computer interaction. 10, 26

HRF hemodynamic response function. 67, 75, 78, 85

IIR infinite impulse response. 40, 52

ISA instantaneous self-assessment. 22, 23, 76

kNN k-nearest neighbours. 1, 11, 25, 32, 44, 51–62, 65, 67, 69, 70, 76, 98, 99

LDA linear discriminant analysis. 1, 11, 27, 31, 32, 44, 51–61, 63, 65, 67, 69, 70, 74–76, 93, 94, 98–100

LSTM long short-term memory. 1, 11, 32, 44, 45, 51–53, 56, 57, 61, 62, 65, 67–69, 76, 98, 99

MBLL modified Beer-Lambert law. 39, 52

MEG magnetoencephalography. 7, 23

NASA-TLX NASA task load index. 21–23

NIRS near-infrared spectroscopy. 7

PFC prefrontal cortex. 17, 24, 25, 38, 40, 67, 71, 72, 79

RNN recurrent neural network. 29, 45

ROC receiver operating characteristic. 30

SVC support-vector classifier. 11, 44, 51–61, 63, 65, 67–70, 76, 98, 99

SVM support vector machine. 1, 27, 29, 32, 44, 52, 78

TDDR temporal derivative distribution repair. 19, 40, 52, 64, 67, 72, 79, 85

Chapter 1

Introduction

1.1 Introduction of the topic

"The ability to monitor, continuously and noninvasively, oxygen sufficiency and circulatory parameters holds promise for a number of applications" – there was the statement made by Frans F. Jobsis in 1977 in a paper that introduced to the world a new neuroimaging technique that would be called functional near-infrared spectroscopy (fNIRS) [82]. It came to complement a range of existing non-invasive methods to measure brain activity such as electroencephalography (EEG), magnetoencephalography (MEG), and functional magnetic resonance imaging (fMRI). fNIRS has gained interest in the neuroscience community lately thanks to its portability, relative tolerance to motion artefacts, and overall ease of use in real-life conditions, especially for the study of mental workload. With the emergence of machine learning, and one of its now most popular sub-disciplines called deep learning, applied to a whole range of disciplines in the last decade, it is only natural to see it applied to fNIRS data too. However, machine learning with fNIRS, and more specifically deep learning with fNIRS, is still currently in its infancy and more research is needed to mature its uses and methods.

We saw machine learning develop tremendously in some fields of application such as computer vision and natural language processing, which have actually contributed to advancing the theory of those techniques quite significantly. This has been in great part made possible by the exponential development of modern computing capabilities which have enabled the blooming of deep learning. It enabled to process larger and larger datasets, naturally aggregated with our society's growing digitalisation.

However, neuroimaging data has its complexities and is found in a much more limited quantity, because as opposed to computer vision and natural language processing for which data can be uploaded and annotated easily with everyday life devices, neuroimaging data can only be collected with rather pricey and often inconvenient equipment, that may require specific expertise to be used. Furthermore, neuroimaging data can be a sensitive type of data to share since it is subconsciously generated and can in some cases be considered as health data. The limited access to data and complexity of pattern recognition is especially true for fNIRS data, and this is why machine learning with fNIRS remains full of challenges.

In this thesis, we will explore machine learning with fNIRS, its standards and best practices, its benchmarks on popular tasks, and its novel approaches, under the prism of mental workload as a key domain of application with fNIRS, and more specifically by having in mind concrete applications to mental workload assessment with brain-computer interface (BCI)s.

1.2 Functional near-infrared spectroscopy

Near-infrared spectroscopy (NIRS) is a technology that was first described by Frans F. Jobsis in 1977 [48, 82]. In this paper, Jobsis explains that near-infrared light can be used to monitor

oxyhemoglobin *in situ*, in a continuous way. Hemoglobin is a protein responsible for transporting oxygen in the blood, generally found in two most common forms: oxyhemoglobin when oxygen is bound to the heme part of the molecule and deoxyhemoglobin when it is not. Those two forms are called chromophores because they absorb light. Jobsis shows that light of wavelengths in near-infrared between 700 and 1300 nm is effectively transmitted through biological materials. Thanks to the modified Beer-Lambert law introduced by Delpy et al. in 1988 [44], it is possible to link the amount of absorbed and scattered light in the biological material to the concentrations in oxy- and deoxyhemoglobin. In the end, thanks to fNIRS it is possible to measure variations of oxygenation in the different areas of the brain, which is very interesting to study brain activity. We will describe more thoroughly how fNIRS leverages the interaction of light with biological tissue to measure changes in oxy- and deoxyhemoglobin in Section 2.1.

An fNIRS device is typically composed of a set of what is called optodes, some of them being emitters (also called sources) and others detectors (Figure 1.1). The light path is described as curved because of diffusion and light is attenuated between the emitter and the detector [145]. By measuring the difference of light intensity between the detector and the source, it is then possible to deduce the concentration in chromophores in between thanks to the modified Beer-Lambert law [44]. The emitter-detector distance is typically 3 cm in adults and enables to measurement of hemoglobin concentrations up to 1.5 cm deep [24, 119, 151]. We call the brain within this depth range the neocortex.

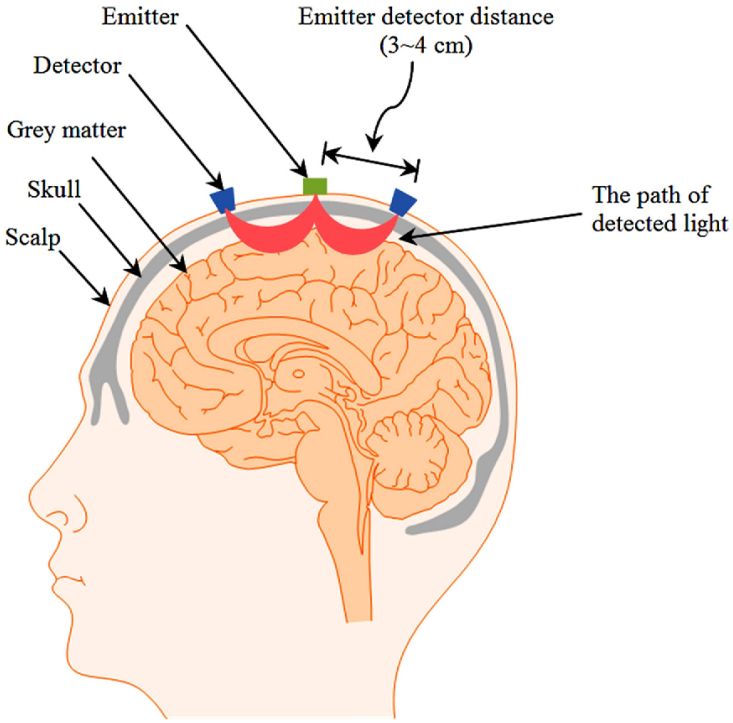

Figure 1.1: Principle of fNIRS *(Source: Kumar et al., 2007)*

In order to measure 2 chromophores, it is required to use at least two different wavelengths in near-infrared. Efforts have been made to find the best combination in order to separate those two chromophores better [17]. It appears that one should be less than 765 nm and one greater than 830 nm [16].

Multiple types of fNIRS exist but the most popular is continuous wave spectroscopy which enables the measurement of oxy- and deoxyhemoglobin changes in a continuous fashion by using light emitted at a constant intensity [22, 150]. This is to differentiate from the less common frequency domain fNIRS emitting frequency-modulated light and time domain fNIRS emitting short pulses of light, which are both usually less affordable.

fNIRS is an interesting tool because concentrations in oxy- and deoxyhemoglobin reflect oxygen consumption in the brain and changes in cerebral blood flow elicited by neuronal activation [135, 150]. This hemodynamic response is referred to as neurovascular coupling [43]. Neurovascular coupling can be evoked by a stimulus or a task, but can also be spontaneous, related to the resting-state brain activity [150]. A parallel is often drawn between the hemodynamic response with fNIRS and the blood oxygenation level-dependent (BOLD) response which is a well-established concept in fMRI [26, 159]. Thanks to studies with both fNIRS and fMRI it has been shown that the BOLD response in fMRI is indeed well correlated with measures of oxy- and deoxyhemoglobin with fNIRS [37, 76, 151, 160].

1.3 Brain-computer interfaces

Neuroimaging data, and in our case fNIRS data, can be used in different ways and for different purposes in research. We can describe two main approaches which are data analysis, and BCI. They can be decomposed into different steps, described in Figure 1.2, some of them being shared by the two approaches.

One can simply perform offline analysis of brain data recordings acquired beforehand by neuroimaging, which is very common in cognitive neuroscience research for example. This typically involves steps including signal acquisition, signal processing, feature extraction, and then statistical inference in order to test a hypothesis, for example comparing brain signals between different experimental conditions (sequence *1.*, *2.*, *3.*, *4.* in Figure 1.2).

On the other hand, neuroimaging data can be interpreted in real-time (online) and used as an input modality of a computer program to perform an action or simply provide feedback: this forms a BCI [32]. It typically involves steps including signal processing of a subject's real-time neuroimaging data, feature extraction, and inference based most commonly on classification. This inference can then simply be used for the purpose of having direct feedback on the brain activity: this is called neurofeedback (sequence *1.*, *2.*, *3.*, *4.*, *6.* in Figure 1.2). Alternatively, it can also be used to execute an action, for example, controlling a computer user interface or an assistive device (eg. wheelchair or prosthesis); the subject can observe their actions, which therefore also provides feedback (sequence *1.*, *2.*, *3.*, *4.*, *5.*, *6.* in Figure 1.2).

We can distinguish multiple types of BCI [180]: *active* BCIs are systems in which the users intentionally attempt to control their brain activity to be interpreted by the interface; *reactive* BCIs represent interfaces in which the interpreted brain activity is evoked and in response to an event presented to the users; *passive* BCIs are systems in which the brain activity interpreted by the system is not voluntarily controlled by the users.

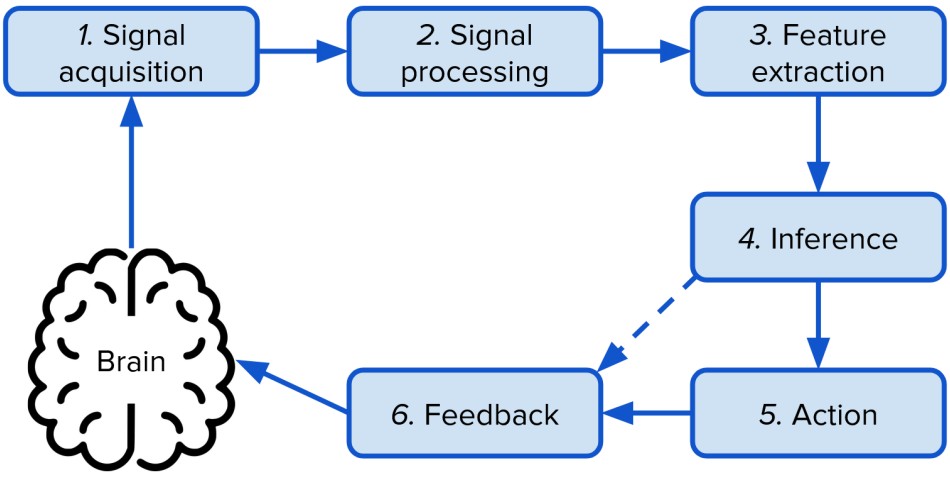

Figure 1.2: Neuroimaging data analysis and BCI diagram

In this thesis, we will mainly be interested in some of the steps of the BCI pipeline: signal processing, feature extraction, and more extensively, inference using classification.

Different algorithms can be used for the classification of fNIRS data, but machine learning is by far the most widespread method [119]. Indeed, machine learning models can be trained on pre-recorded data, in order to then be used for online classification of real-time data in the context of a BCI.

1.4 Application to mental workload

Mental workload, often called cognitive load, is a concept described in 1991 by Eggemeier et al. [46] with the following definition: *"Mental workload refers to the portion of operator information processing capacity or resources that is actually required to meet system demands"*. Put simply, it can be described as the amount of mental resources used when performing a task. This concept is highly related to attention and working memory which represents the short-term memory that can be used and processed immediately to perform a task.

Mental workload is very popular in the context of ergonomics and human factors, and with digital technologies taking more and more space in our lives it is also very interesting in the context of human-computer interaction (HCI).

Cognitive load can be a stake of safety in a context where error cannot be accepted. Indeed having a high mental workload can lead to committing mistakes on a task at hand that can potentially be dramatic in the case of safety-critical environments. This is what Young et al. explain with the concept of workload *redlines* [175], describing the fact that too high or too low mental workload can be problematic.

1.5 Problem statement and research questions

Seeing the importance of mental workload assessment, we can understand why classifying mental workload in real-time can be important, and for this purpose, fNIRS BCIs could be a solution, if deployed in a responsible way. However, while such systems are emerging, they are currently not ready for deployment in real-life scenarios due to their lack of reliability, and algorithms need to be improved in order to make fNIRS BCIs better. This is a challenge in itself, but furthermore, the complexity and variety of methods for evaluating the performance of fNIRS BCI algorithms make it hard to build on existing research. We therefore ask the following research questions that this thesis will aim at answering in its following chapters:

- *RQ1*: How can we make the rigorous development, evaluation, and comparison of machine learning approaches for task classification from fNIRS simpler for researchers? → Chapter 3 (and validated Chapters 4, 5, and 6)
- *RQ2*: What are the benchmarks of popular machine learning models on various tasks from open access fNIRS datasets? → Chapter 4
- *RQ3*: Across these benchmarks, what factors influence the machine learning classification accuracy? → Chapter 4
- *RQ4*: How do machine learning approaches tailored to a specific mental workload task compare to baseline benchmarks? → Chapter 5
- *RQ5*: How can using unlabelled segments of the fNIRS recordings help with classification? → Chapter 6

Answering these research questions will facilitate machine learning with fNIRS for researchers in the field, by making the development of such techniques simpler and less time-consuming. It

will also enable establishing objective benchmarks of machine learning models for fNIRS BCI applications with insights regarding factors that could influence their performance. Finally, it will shed light on new machine learning directions for fNIRS applications.

1.6 Summary of the contributions

In answering the research questions, this thesis made contributions that can be divided into two main parts.

The first set of contributions is a framework for machine learning with fNIRS called *BenchNIRS*. This framework is an open-source Python software and enables to:

- load open-access fNIRS datasets from the community;
- process fNIRS signal with state-of-the-art methods;
- extract features;
- use a state-of-the-art machine learning methodology to develop new machine learning models including training, optimisation, evaluation, and comparison of models for classification from fNIRS data;
- produce clear visualisations for monitoring training and presenting results.

BenchNIRS enables the use of different approaches and paradigms. This includes the ability to use both supervised and self-supervised learning and to perform transfer learning. It also gives the ability to study different influencing factors on the performance of the machine learning models (time window length, number of training examples, sliding window) with different generalisation goals (subject-independent or subject-specific). It comes with 6 machine learning models pre-implemented, including linear discriminant analysis (LDA), support-vector classifier (SVC), k-nearest neighbours (kNN), artificial neural network (ANN), convolutional neural network (CNN) and long short-term memory (LSTM), and provides example scripts for the evaluation of machine learning models with statistical analysis. This framework of more than 3,500 lines of code enables simplified machine learning for fNIRS and makes it much less time-consuming to develop new models by enabling the implementation of a whole pipeline in a dozen lines of code. It makes machine learning accessible for fNIRS researchers with little machine learning experience, and makes fNIRS accessible for machine learning researchers with little fNIRS experience, bridging the gap between those two advanced and technical fields. This set of contributions is introduced in Chapters 3 and 6, and validated in Chapters 4, 5 adn 6.

The second set of contributions consists of providing benchmarks of different machine learning approaches for classification from fNIRS on various tasks. This includes:

- the evaluation of various baseline machine learning models including LDA, SVC, kNN, ANN, CNN and LSTM;
- the evaluation of the influence of different factors on the machine learning performance including the size of the time window, the number of training examples, and the use of a sliding window.
- the evaluation of the influence of different generalisation goals including subject-independent classification and subject-specific classification;
- the evaluation of the tailoring of machine learning models on specific mental workload tasks and datasets;
- the evaluation of different machine learning paradigms including supervised learning, self-supervised learning, and transfer learning.

This contributes to giving a starting point for comparison of many different machine learning approaches with fNIRS. It also provides indications as to what models and approaches are best suited to different use cases, which is useful when it comes to choosing an algorithm for a specific BCI application. This set of contributions is presented in Chapter 4, 5 and 6.

1.7 Publications

1.7.1 Preamble to the thesis

- Conference article: "Exploring machine learning approaches for classifying mental workload using fNIRS data from HCI tasks". J Benerradi, HA Maior, A Marinescu, J Clos, ML Wilson. *Proceedings of the Halfway to the Future Symposium 2019*. November 2019.

1.7.2 Main publications

- Extended abstract and talk: "Benchmarking framework for machine learning with fNIRS". J Benerradi, J Clos, A Landowska, MF Valstar, ML Wilson. *Neuroergonomics conference*. September 2021.
- Journal article **(edited into Chapters 3 and 4)**: "Benchmarking framework for machine learning classification from fNIRS data". J Benerradi, J Clos, A Landowska, MF Valstar, ML Wilson. *Frontiers in Neuroergonomics*. March 2023.
- Poster (best poster award at fNIRS UK 2023): "BenchNIRS: benchmarking framework for machine learning with fNIRS". J Benerradi, J Clos, A Landowska, MF Valstar, ML Wilson. *fNIRS UK 2023*. September 2023.

1.7.3 Software contributions

- BenchNIRS[1] (featured on the OpenfNIRS website[2])
- NIRSimple[3]
- MNE contributions[4]

[1] `https://gitlab.com/HanBnrd/benchnirs`
[2] `https://openfnirs.org/software/`
[3] `https://github.com/HanBnrd/NIRSimple`
[4] `https://github.com/mne-tools/mne-python/pulls?q=is%3Apr+author%3AHanBnrd`

Chapter 2

Literature Review

In this chapter, we will explain in detail key scientific concepts that are the building blocks of this thesis, interested in machine learning for fNIRS BCIs through the scope of applications in mental workload assessment, and draw a picture of the related literature around it.

We will more specifically dive into the science behind fNIRS and its signal processing; we will define key concepts related to mental workload, its elicitation, measurement, and influence on physiology; and finally, we will take a tour of machine learning, its applications for making BCIs and its limitations.

2.1 Functional near-infrared spectroscopy

2.1.1 Pre-processing

fNIRS relies on the interpretation of the attenuation of light emitted by the sources in brain tissue. This is measured with the detectors for each source-detector pair called *channel*. What we call pre-processing with fNIRS consists in converting the light intensities measured with the detectors into a spatio-temporal evolution of oxyhaemoglobin (HbO) and deoxyhaemoglobin (HbR).

As the light emitted by a source diffuses in all directions, if every source was emitting at the same time, it would be hard to identify which source is received by each detector to constitute the source-detector channels. Therefore, one of the simplest methods used with continuous wave fNIRS hardware consists in flashing the sources one at a time (or in small groups), so that based on the moment at which each detector receives light, it is possible to retrace which source is received by detectors and then obtain the light attenuation for each channel specifically. This principle is called time-multiplexing [176].

The light attenuation is expressed with the optical density, sometimes also called absorbance, given by the following formula:

$$OD = \log_{10}(\frac{I_0}{I}) \tag{2.1}$$

With:
- OD the optical density (dimensionless);
- I_0 the incident optical intensity;
- I the transmitted optical density.

When light goes through an absorbing medium containing chromophores, we know thanks to the original Beer-Lambert law that the optical density is proportional to the pathlength of the light in the medium and the concentration in chromophores (Figure 2.1), according to the following formula:

$$OD = \varepsilon \times l \times C \tag{2.2}$$

With:
- OD the optical density (dimensionless);
- ε the molar extinction coefficient in $mol^{-1}.L.cm^{-1}$;
- l the pathlength of the light in the medium in cm;
- C the concentration of the chromophore in $mol.L^{-1}$.

The coefficient of proportionality is called the extinction coefficient ε. It depends on the wavelength of the light that goes through the medium and is specific to each chromophore. Indeed, each chromophore has a different absorption spectrum, which corresponds to the amount of attenuation depending on the wavelength. Those absorption spectra are usually measured empirically for each chromophore and can be found in the literature. For example, HbO and HbR absorb light differently and each has specific tables of extinction coefficients [36, 139, 172, 183]. The most popular extinction coefficient dataset for fNIRS is the one from Wray et al., 1988 [172].

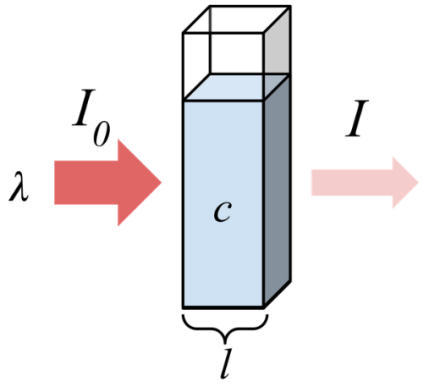

Figure 2.1: Path of the light in a cuvette for which the Beer-Lambert law can be applied simply *(Source: commons.wikimedia.org)*.

The Beer-Lambert law is very useful in controlled settings, however, biological tissues are complex and present different layers of varying density and refractive indices leading to scattering. The brain is therefore both an absorbing medium due to the presence of chromophores and a scattering medium due to irregularities and layers with different properties. This is why the modified Beer-Lambert law was proposed by Delpy et al. in 1988 [44] to estimate chromophore concentrations with fNIRS. It introduces a new coefficient called the differential pathlength factor which depends on the wavelength to account for the scattering of the light in the brain. The modified Beer-Lambert law is also described with a geometry factor that accounts for the geometry of the measurement [75]. The path of the light in the brain can be seen in Figure 2.2. We then have the following formula:

$$OD = \log_{10}(\frac{I_0}{I}) = \varepsilon \times d \times DPF \times C + G \tag{2.3}$$

With:
- OD the optical density (dimensionless);
- I_0 the incident optical intensity;
- I the transmitted optical density;
- ε the molar extinction coefficient in $mol^{-1}.L.cm^{-1}$;
- d the distance between the source and the detector in cm;
- DPF the differential pathlength factor (dimensionless);
- C the concentration of the chromophore in $mol.L^{-1}$;
- G the geometry factor (dimensionless).

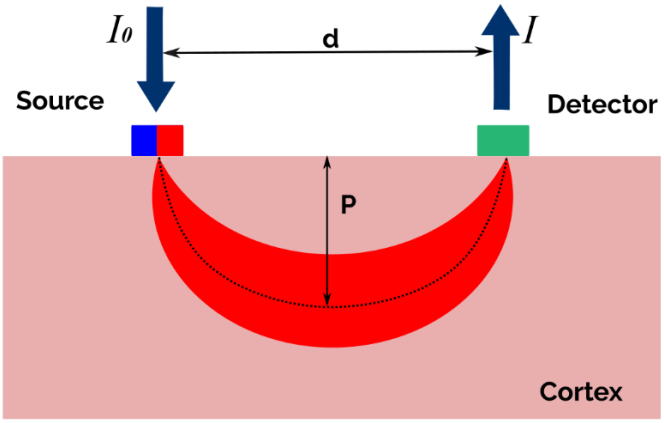

Figure 2.2: Path of the light in the brain with fNIRS for which we can apply the modified Beer-Lambert law *(Source: newmanbrain.com)*.

However, the geometry factor is not measurable with fNIRS. This is why we get concentration changes out of fNIRS measurements, which are usually the difference between measurement at time t and a reference measurement, which can be the average intensity over the whole measurements or can be recorded on a dedicated baseline. This can be archived because G is constant over time during an fNIRS recording. By using this method, we do not need G anymore, and as a consequence, I_0 the incident intensity is not necessary as well. We therefore obtain the following formula:

$$\Delta OD = OD_t - OD_{ref} = \log_{10}(\frac{I_{ref}}{I_t}) = \varepsilon \times d \times DPF \times \Delta C \tag{2.4}$$

With:
- ΔOD the optical density change (dimensionless);
- OD_t the optical density at time t (dimensionless);
- OD_{ref} the reference optical density (dimensionless);
- I_t the transmitted optical intensity at time t;
- I_{ref} the reference transmitted optical density (averaged on a baseline period for example);
- ε the molar extinction coefficient in $mol^{-1}.L.cm^{-1}$;
- d the distance between the source and the detector in cm;
- DPF the differential pathlength factor (dimensionless);
- ΔC the concentration change of the chromophore in $mol.L^{-1}$.

This previous equation is the one that describes the relation between the optical density change and the concentration change for one chromophore. The human blood however contains 4 major chromophores that absorb light in the near-infrared: oxyhemoglobin, deoxyhemoglobin, oxymyoglobin and deoxymyoglobin; but myoglobin is mostly present in muscles so it can be neglected in the brain. With the 2 chromophores detected in the brain, we need to account for the optical density change due to HbO and the optical density change due to HbR. It is then required to use 2 different wavelengths to obtain a system of 2 equations with 2 unknown variables, which can be solved easily with matrix operations. The system of equations is solved the following way:

$$\begin{cases} \Delta OD_{\lambda_1} = \varepsilon_{HbO,\lambda_1} \times d \times DPF_{\lambda_1} \times \Delta C_{HbO} + \varepsilon_{HbR,\lambda_1} \times d \times DPF_{\lambda_1} \times \Delta C_{HbR} \\ \Delta OD_{\lambda_2} = \varepsilon_{HbO,\lambda_2} \times d \times DPF_{\lambda_2} \times \Delta C_{HbO} + \varepsilon_{HbR,\lambda_2} \times d \times DPF_{\lambda_2} \times \Delta C_{HbR} \end{cases} \tag{2.5}$$

$$\begin{bmatrix} \Delta OD_{\lambda_1} \\ \Delta OD_{\lambda_2} \end{bmatrix} = \begin{bmatrix} \varepsilon_{HbO,\lambda_1} \times d \times DPF_{\lambda_1} & \varepsilon_{HbR,\lambda_1} \times d \times DPF_{\lambda_1} \\ \varepsilon_{HbO,\lambda_2} \times d \times DPF_{\lambda_2} & \varepsilon_{HbR,\lambda_2} \times d \times DPF_{\lambda_2} \end{bmatrix} . \begin{bmatrix} \Delta C_{HbO} \\ \Delta C_{HbR} \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} \Delta C_{HbO} \\ \Delta C_{HbR} \end{bmatrix} = \begin{bmatrix} \varepsilon_{HbO,\lambda_1} \times d \times DPF_{\lambda_1} & \varepsilon_{HbR,\lambda_1} \times d \times DPF_{\lambda_1} \\ \varepsilon_{HbO,\lambda_2} \times d \times DPF_{\lambda_2} & \varepsilon_{HbR,\lambda_2} \times d \times DPF_{\lambda_2} \end{bmatrix}^{-1} . \begin{bmatrix} \Delta OD_{\lambda_1} \\ \Delta OD_{\lambda_2} \end{bmatrix} \tag{2.7}$$

With:
- λ_1 and λ_2 two different wavelengths;
- HbO and HbR the oxyhemoglobin and deoxyhemoglobin respectively.

2.1.2 Sources of noise

Even though fNIRS is a useful technology to measure brain activity, it has some weaknesses as every neuroimaging technology and the quality of signals measured can be affected by different kinds of noise. Sources of noise can be sorted into three categories: instrumental noise, experimental noise, and physiological noise [119].

Instrumental noise is related to the hardware characteristics. It usually originates from the environment, for example with external light, and tends to cause high-frequency constant noise. It can also originate from the gradual movement of the optodes on the scalp in which case it tends to appear as low-frequency drifts [42].

A significant source of experimental noise is due to the subject's movement called motion artefacts. More specifically this usually comes from the movement of optodes relative to the skin or hair. This can happen for example because of head motion. This type of noise appears in the signals in the form of high-frequency spikes or baseline shifts due to rapid movements, but also low-frequency artefacts due to slower head movements for example [78].

Finally, we have the physiological noise originating from different global systemic and local regulatory processes which is often the most prominent source of noise with fNIRS. It corresponds mostly to the heart pulsation around 1 to 1.5 Hz, the respiration around 0.2 to 0.5 Hz, and Mayer waves related to arterial blood pressure around 0.1 Hz, which all affect the sensitivity to functional neuronal signals of interest [89, 119], since fNIRS measures blood oxygenation changes regardless of their cause.

Other than those three types of noise, other factors can also affect the quality of signals, such as skin tones and hair types which may result in light being absorbed differently, so are important to keep in mind [91].

2.1.3 Comparison to other neuroimaging technologies

fNIRS is a very interesting technology as it has the advantage of monitoring both HbO and HbR while fMRI can only measure the BOLD response which focuses on HbR [136]. This allows for more information about oxygen consumption in the neocortex.

fNIRS has a relatively good temporal resolution, of the order of 5 seconds [39], better than fMRI, and can then be suitable for use in real-time, even though this temporal resolution is lower than EEG [122]. Even if fNIRS devices can have high sampling frequencies, often around 25 Hz, the limitation comes from the hemodynamic response itself which takes time to appear after neuronal activation (delay of 1-2 seconds and temporal width of 4-6 seconds [26]).

The spatial resolution is also good, even though it is lower than in fMRI. It is for example better than the spatial resolution of EEG [122].

Moreover, even if motion artefacts remain a source of noise in fNIRS, it is quite tolerant to those compared to other neuroimaging techniques such as EEG [122].

The physical properties of fNIRS make it possible to manufacture portable devices that can be wireless thanks to Bluetooth for example (Figure 2.3) unlike fMRI.

Figure 2.3: Portable Bluetooth fNIRS device (Brite from Artinis) *(Source: artinis.com)*.

The spatial and temporal resolutions make it a tool of choice to study processes such as mental workload [106], especially in the prefrontal cortex (PFC) [102], and this is why many portable devices focus on this specific brain area such as the OctaMon or Brite (Figure 2.3) from Artinis, the fNIRS300 from Biopac and many others.

2.2 fNIRS signal processing

The fNIRS data collected are time series, and it is required to process the signal in order to extract meaningful information from it. For the most part, it consists in removing the different forms of noise in order to extract a better brain response and have the best signal-to-noise ratio.

2.2.1 Filters

One of the most common techniques is filtering. It consists in removing ranges of frequencies that are known not to correspond to any relevant signal for brain activity. Filters can be classified into different categories based on what they achieve. We have high pass filters which remove frequencies below a threshold, low pass filters which remove frequencies above a threshold, and band pass filters which are a combination of the two. Bandpass filter enable to only keep frequencies between two threshold frequencies called cutoff frequencies. By doing so, only frequencies in this range called passband remain. We call stopband the frequencies outside of this passband. Finally, another category is the notch filter which only removes one frequency. This type of filter is often used to remove electrical noise which is typically of a specific frequency, constant over time.

The strength of attenuation of a filter in the stopband, also called gain, can be modified thanks to the order of the filter. The higher the order, the higher the gain, and thus the stronger the attenuation in the stopband.

Different methods exist to perform those filters but one of the most common is the Butterworth filter [134]. It has the advantage of not distorting the signal in the bandpass.

The Butterworth filter is defined by its transfer function H which depends on the frequency at a given order and cutoff frequency [25] according to the following formula:

$$|H_n(j\omega)| = \frac{1}{\sqrt{1 + (\omega/\omega_c)^{2n}}} \tag{2.8}$$

With:
- H_n the transfer function of the filter of order n;
- ω the angular frequency in $rad.s^{-1}$, $\omega = 2\pi f$ with f the frequency in Hz;
- ω_c the angular cutoff frequency in $rad.s^{-1}$, $\omega_c = 2\pi f_c$ with f_c the frequency in Hz.

A bode diagram enables us to visualise the gain depending on the angular frequency for this Butterworth filter as shown in Figure 2.4.

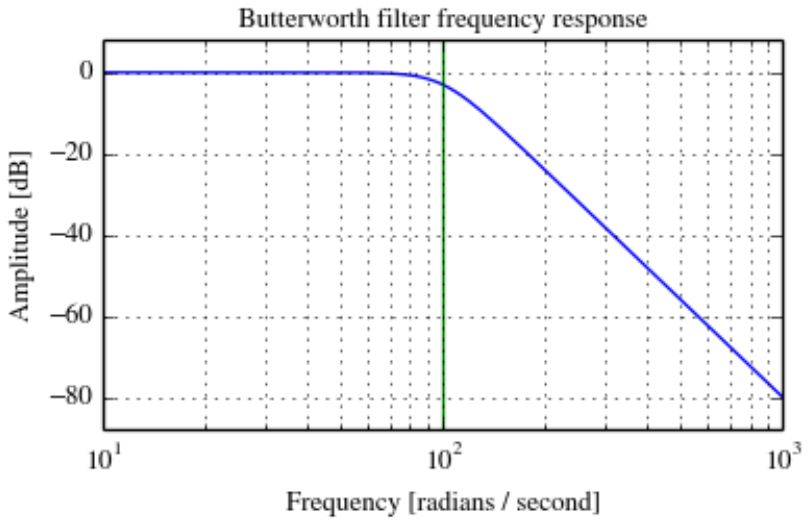

Figure 2.4: Bode plot of a lowpass filter of order 4 with an angular cutoff frequency of 100 r$ad.s^{-1}$ *(Source: docs.scipy.org).*

Bandpass filters are very useful when it comes to removing physiological noise. Indeed, as the hemodynamic response has a frequency of around 0.1 Hz, it can be interesting to keep a frequency range around it and remove everything else. This enables the removal of heart rate and respiration artefacts when cutoff frequencies are for example 0.01 and 0.5 Hz, which is the most popular filter found in the literature [134]. It also enables the removal of slow drifts with low frequency. Unfortunately, this cannot remove Mayer waves which are right in the frequency range of the hemodynamic response.

2.2.2 Motion artefact correction

Correlation based signal improvement

When interested in addressing motion artefacts more specifically, an interesting technique is the correlation based signal improvement (CBSI) [38]. It takes advantage of the fact that measurements of HbO and HbR are usually negatively correlated in the absence of head motion. It generates corrected artificial HbO and HbR signals smoothing out motion artefacts based on negative correlation which improves the signal-to-noise ratio. The system of equations to obtain the corrected HbO and HbR is the following:

$$\begin{cases} x_0 = \frac{1}{2}(x - \alpha y) \\ y_0 = -\frac{1}{\alpha} x_0 \end{cases} \tag{2.9}$$

With:
- x and y respectively the measured HbO and HbR concentration changes;

- x_0 and y_0 respectively the corrected HbO and HbR;
- α the ratio of the standard deviation in HbO and HbR on a chosen window.

This approach can be very interesting, especially for real-time applications when applied with a sliding window of around 30 seconds.

Temporal derivative distribution repair

Another technique often used in order to remove motion artefact is the temporal derivative distribution repair (TDDR) [49]. Assuming that fluctuations related to motion artefact are large and infrequent as opposed to the rest of the contributions to signal fluctuations (hemodynamic response reflecting brain activity and other hemodynamic activity related to systemic regulations), this approach consists in correcting the signal by reducing the weights of abnormally large fluctuations. It does so by computing the temporal derivative of the signal (representing the speed of change of the signal amplitude), to then correct it with a weighting function, placing lower weight on larger deviations, defined as follows:

$$w_t = \begin{cases} (1 - d_t^2)^2 & if \quad |d_t| < 1 \\ 0 & otherwise \end{cases} \tag{2.10}$$

With:
- w_t the weighting function (function of time t);
- d_t the scaled residual at time t (scaled difference between the weighted mean of the fluctuations and the observed value).

With this technique, it is possible to attenuate relatively well the signal's spikes and baseline shifts, which cannot otherwise be corrected with standard filtering techniques.

Deep learning

Recent advances in deep learning have shed light on new approaches for motion artefact removal. For example, Gao et al. [55] propose an approach using a denoising autoencoder neural network model in order to reconstruct clean fNIRS signals from noisy data. This model was trained with simulated data on a regression problem consisting in reconstructing a clean signal from the signal to which artificial noise has been added. This trained model can then be used on experimental data in order to remove real motion artefacts.

2.2.3 Short separation channels

More recently, a technique that has gained interest is the regression with short separation channels (Figure 2.5). Indeed, normally the separation between the emitter and the detector is around 3 cm to access concentration changes in the neocortex. The shorter the distance the shallower it measures. This principle is used with short channels: we measure concentration changes in shallower structures that we know do not represent brain activity, namely the scalp. The activity in the scalp contains all the systemic noise which is not interesting when it comes to measuring brain activity. Therefore, we can remove this systemic noise by regressing the short separation channel measurements to the long separation channel measurements. Only remains the concentration changes related to the brain activity. This technique of short separation channels works very well but requires having those specific channels on the fNIRS device which is not always the case, especially for old equipment.

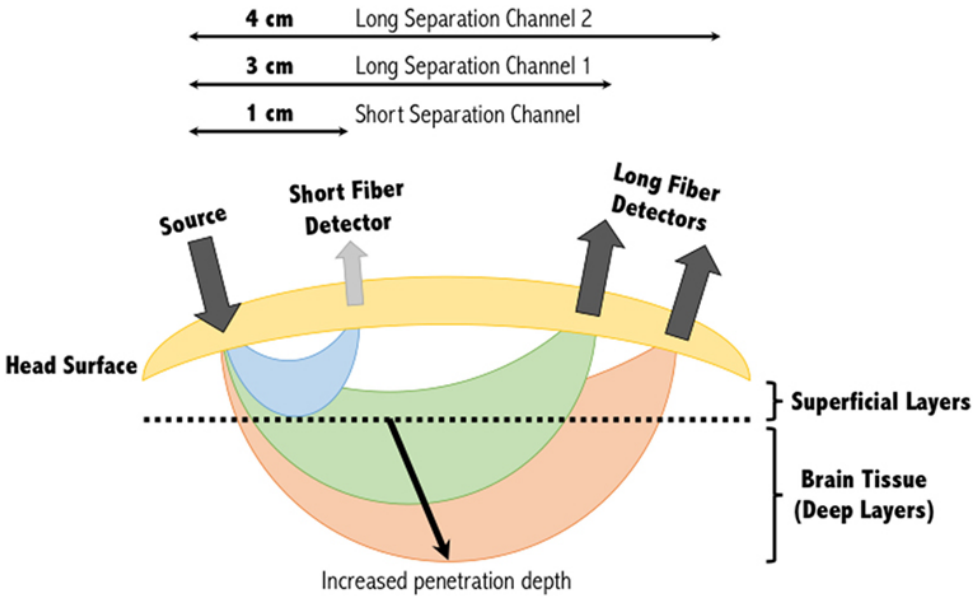

Figure 2.5: Short separation channels *(Source: Rupawala et al., 2018)*.

2.3 Mental workload

2.3.1 Tasks eliciting mental workload

Multiple tasks have been developed to elicit mental workload and are used extensively in neuropsychology experiments for this purpose.

n-back

n-back is probably one of the most popular tasks to elicit mental workload and has been used in many neuropsychology studies [125]. It consists in presenting a sequence of stimuli, which can be images or sounds, and the task for the subject is to indicate when the stimulus currently presented (target) is the same as the one presented n stimuli before. When n increases, the difficulty increases, and levels usually range between $n = 1$ and $n = 3$ during experiments. The n-back task can use various types of stimuli such as visual or auditory stimuli and can use verbal stimuli such as letters or words or nonverbal stimuli such as shapes, faces, or pictures. Some studies consist in monitoring the identity of the stimulus whereas others consist in monitoring the location of it. An example of numerical 2-back is presented in Figure 2.6.

Mental arithmetic

The second task often used consists in doing arithmetic calculations mentally. This can be additions, subtractions, multiplications, or divisions. A task that is commonly used is backward counting with subtraction of a specific number every time [67, 107, 119].

Word generation

Finally, another popular task to elicit mental workload is word generation [67, 155]. This consists in thinking of words beginning with a given letter as fast as possible.

2.3.2 Measuring mental workload

Measuring mental workload can be done by multiple means. Some of the most popular are subjective questionnaires while other measures are more objective.

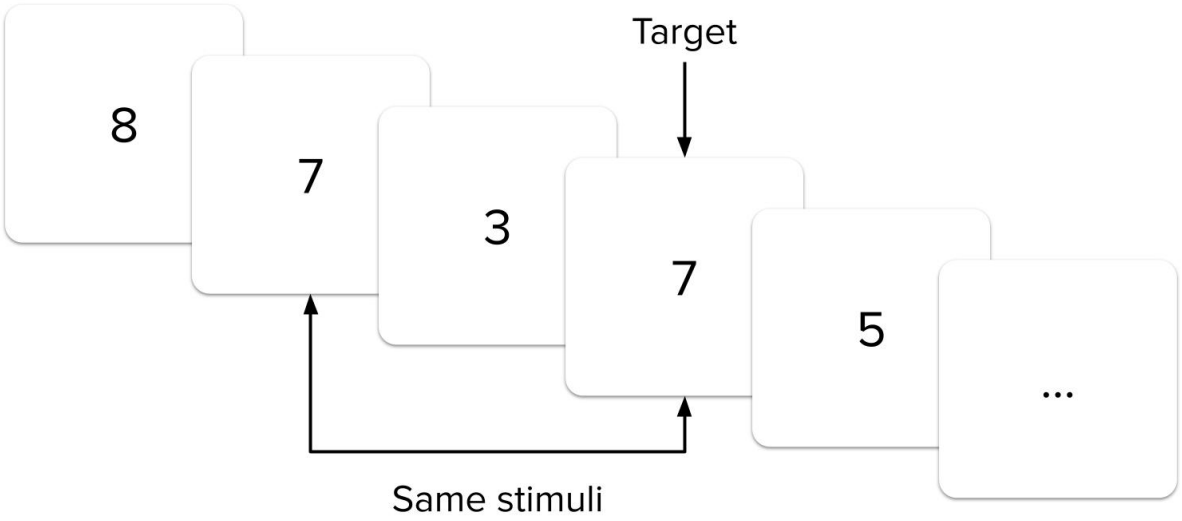

Figure 2.6: Example of numerical n-back task (2-back).

NASA Task Load Index

The NASA task load index (NASA-TLX) is a questionnaire developed by NASA for subjective assessment of the perceived workload [63]. It assesses multiple characteristics: mental demand, physical demand, temporal demand, performance, effort, and frustration. Each of these category is evaluated on a scale by subjects. This questionnaire is usually answered after the tasks. An example can be seen in Figure 2.7.

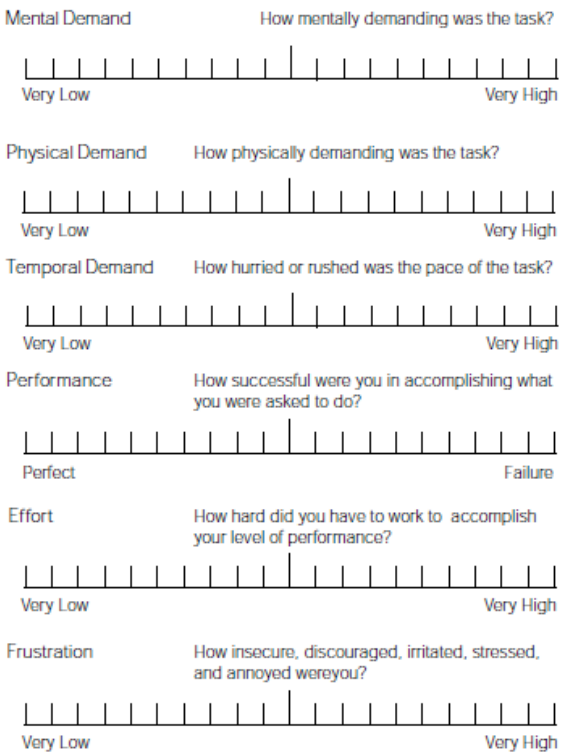

Figure 2.7: NASA-TLX questionnaire *(Source: humansystems.arc.nasa.gov)*.

Instantaneous self-assessment

Another way to measure mental workload is the instantaneous self-assessment (ISA) technique. It has been developed by NATS in order to assess the mental workload of air traffic controllers [83]. With this method, subjects report regularly their mental workload level on a pre-established scale from 1 to 5. The meanings of each rating can be found in Figure 2.8. Those ratings are usually given verbally by subjects while they perform a task.

Level	Workload Heading	Spare Capacity	Description
5	Excessive	None	Behind on tasks; losing track of the full picture
4	High	Very Little	Non essential tasks suffering. Could not work at this level very long.
3	Comfortable Busy Pace	Some	All tasks well in hand. Busy but stimulating pace. Could keep going continuously at this level.
2	Relaxed	Ample	More than enough time for all tasks. Active on ATC task less than 50% of the time.
1	Under– Utilised	Very Much	Nothing to do. Rather boring.

Figure 2.8: ISA workload categories *(Source: Kirwan et al., 1997)*.

Physiological measures

Methods such as the NASA-TLX or the ISA however have some bias because they are actually subjective. Indeed, even if the questions and criteria are pre-established, they are answered by subjects based on their perception of their mental workload which can sometimes be different from their actual mental workload. This is why a lot of effort in the scientific community has been made lately to find objective measures of mental workload, and physiological measurements are interesting for this purpose as they are for the most part hardly controllable intentionally. Brain activity especially stands out as a strong candidate among those physiological measurements as it enables the measurement of mental workload directly at the source.

Physiology can also enable researchers to measure the mental workload passively without any action from the subject that could distract them from the task. For example ISA creates the additional task of reporting the mental workload to the original task which can impact performance as Tattersall and Foord show [161]. More precisely, the fact of asking participants while they perform a task can increase their mental workload [131]. Also, adding mental workload reporting as a secondary task could lead to adaptive strategy changes to the main task which is not what one would want to study subjects' response to that main task [33].

2.3.3 Influence of mental workload on physiology

Physiology can be affected by changes in mental workload in different ways.

Pupil diameter

First of all, pupil diameter has been shown to be an interesting physiological measurement related to mental workload. In a constantly lit environment, pupil dilatation or constriction has

been observed in response to changes in mental workload. A review paper by Beatty gives insight into research studying the link between mental workload and pupil diameter [9]. Iqbal et al. for example show that pupil diameter correlates positively with the cognitive load [77]: when the load increases pupil dilatation can be observed and conversely, when it decreases pupil constriction can be observed. Another work by Marinescu et al. shows a strong correlation between the ISA scores used to evaluate mental workload and the pupil diameter [109].

Facial skin temperature

Another interesting physiological modality related to mental workload is facial skin temperature. In addition to pupil diameter Marinescu et al. show that facial temperature is correlated to the ISA rating and NASA-TLX score [109]. Pinti et al. also show that measurements from functional infrared thermal imaging are modulated by cognitive tasks [133]. More precisely, Or and Duffy show that the nose temperature has an important correlation with mental workload [124]: a nose temperature drop is observed with high mental workload.

Heart rate and respiratory rate

Researchers in the field of mental workload have also shown that heart rate is an interesting physiological measurement related to mental workload. For example, when subjects were performing air traffic control tasks, Tattersall and Foord showed that the ISA was correlated to heart rate variability [161]. Also, the respiratory rate has also been investigated and is shown to be related to the mental workload [109] as well.

Electrodermal activity

Electrodermal activity (EDA) also known as skin conductance or galvanic skin response is yet another interesting physiological parameter of the body related to mental workload. Interesting research by Shi et al. shows that EDA is related to cognitive load [152]. Indeed, in their study, galvanic skin response increases as cognitive load increases. Another study by Foy and Chapman on drivers' mental workload highlights that skin conductance was affected by the road type and increased with increasing mental workload.

Brain activity

The physiological measurement that probably makes the most sense in the case of mental workload is brain activity. Indeed, in response to an increase in mental workload demand, the brain has to solicit its resources in order to adapt to it.

Many functional neuroimaging techniques exist in order to study brain activity. Some of them focus on the electromagnetic properties of the brain activity such as EEG and MEG while others focus on blood oxygenation such as fMRI, and fNIRS as we have seen in 2.1.1. Neuroimaging techniques can also be invasive, in the way that they require a surgical procedure to be used, or non-invasive, most of the time with measurement on the scalp.

Recently fNIRS is a type of neuroimaging that has gained interest in order to study mental workload. Indeed, it has been shown that brain activity measured with fNIRS correlates with the difficulty of working memory tasks such as n-back [171]. Authors such as Peck et al. or Maior et al. also observe for example a negative correlation between the concentration in HbR and results from the NASA-TLX on working memory tasks [107, 127].

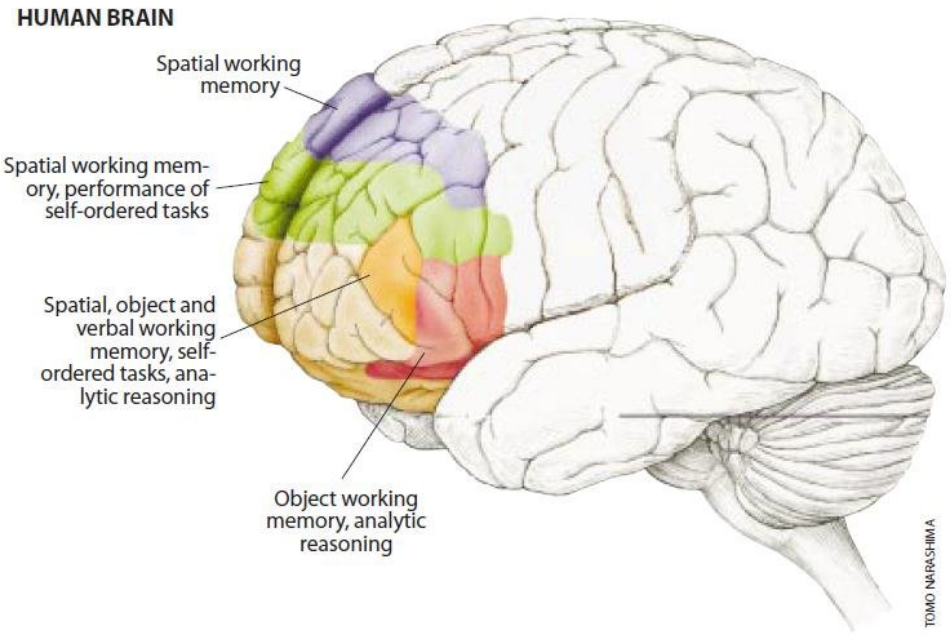

Figure 2.9: Prefrontal cortex of the brain *(Source: Beardsley, 1997).*

Research in the neuroscience of mental workload provides evidence that a key area of the brain to study this kind of activity is the PFC (Figure 2.9). Molteni et al. show a brain oxygenation increase in the frontal brain with a working memory load increase, and furthermore, the activity is lateralised [114]. Nozawa et al. show for example that the left PFC is associated with spatial memory while the right PFC is associated with verbal working memory [123]. Indeed it is shown that different types of tasks to elicit mental workload activate different areas in the PFC. For example, Hoshi et al. show that activation of the ventrolateral PFC happens during n-back working memory tasks, while random number generation activates the PFC differently [72]. They conclude by saying that the PFC is the central executive of working memory. Causse et al. show that besides being observed in laboratory tasks, changes in the PFC in response to mental effort are also observed in ecological settings [29].

2.3.4 Mental workload assessment from physiological data

In opposition to just being observed in response to various tasks, physiology can be used as a way to assess mental workload. This process usually relies on machine learning, in which a labelled dataset is recorded to train a classifier that could predict mental workload in real-time. The dataset is usually labelled using task difficulty or level, or the subjective ratings of mental workload made by the subjects as seen in 2.3.2.

Assessment from physiological indicators

Most of the physiological indicators linked to mental workload described previously in 2.3.3 can potentially be used to assess mental workload.

For example, Fridman et al. [52] used pupil diameter in order to predict mental workload in driving settings. They use a deep learning approach with camera recordings. It enables the detection of the pupil within the frame and obtaining the pupil diameter which can then be used to predict the mental workload of drivers.

We also find cases where the facial skin temperature is used to infer mental workload for example with the work of Or and Duffy [124]. It would then be possible to asses mental workload with an infrared thermal imaging camera.

Haapalainen et al. were able to distinguish between low and high levels of cognitive load thanks to electrocardiogram median absolute deviation and median heat flux measurements [62]. The tasks on which they predict cognitive load are based on visual perception and cognitive speed which are relevant to assess ubiquitous computing systems and divided attention.

However it is not always easy to be sure that those physiological indicators are directly related to mental workload, and each of the following modalities has some disadvantages. Indeed other factors can influence the pupil diameter such as the ambient lighting conditions which are required to be constant if one wants to have a good indication of mental workload. In the same way, heart and respiration rate, facial skin temperature, or EDA can be modified with physical activity, regardless of any mental workload change. Other confounding factors can also enter into play, for example, research by Nickel and Nachreiner show that heart rate variability may not be a good indicator of cognitive load but rather an indicator of time pressure [121].

It is then more relevant to measure the mental workload right from where it originates, namely the brain. Those reasons make brain measurements more suited for mental workload assessment as they can measure the process at the source.

Brain-computer interfaces

Neuroimaging techniques can be used to measure brain activity in areas of the brain that are known to be related to mental workload, namely the PFC. Using neuroimaging as a way to have feedback is called neurofeedback, and this is one type of BCI as seen in 1.3.

Due to their good temporal resolution and portability, EEG and fNIRS are devices of choice for BCIs with a short response time.

Work has been done with EEG to work towards the creation of neurofeedback interfaces on mental workload. For example, Lan et al. show that it is possible to classify cognitive load offline thanks to EEG brain recordings [92]. After a feature extraction based on frequency analysis, they were able to classify levels of cognitive load on n-back working memory tasks with Gaussian mixture models or kNN. Appriou et al. tried to achieve classification on the same kind of tasks from EEG by comparing different models such as common spatial pattern (CSP), Riemannian geometry, and a shallow CNN [5].

Even though EEG is the most popular brain imaging technique for BCIs, one does not necessarily need such a high temporal resolution (which can be of the order of milliseconds with EEG) in cases where making inferences with a slight delay (a couple of seconds) can still be suitable. In this context, research into continuous wave fNIRS is increasing due to its greater tolerance to user's motion compared to EEG [122] which signals are often hardly recoverable in the presence of body and head movements. This is especially useful when working in naturalistic settings or with infants for example. Furthermore, fNIRS is characterised by a lower temporal resolution but is capable of higher spatial resolution than EEG [122], and the frequencies of interest with fNIRS (hemodynamic response under 1 Hz) are usually lower than the ones studied with EEG [32, 141]. FNIRS still faces challenges to be used reliably in real-life conditions, but more and more experiments are working towards this goal [132], with some studies focusing for instance on walking [167] and climbing [28]. FNIRS can be used for active BCIs consisting of prolonged and sustained brain activity for example, but is currently mostly used for passive BCIs [180, 181] due to the 1 to 2 seconds delay in cerebral blood flow and a peak response 4 to 6 seconds after a stimulus [26].

A lot of tasks have been used in lab settings in order to advance BCI research. The first category falls under the *active BCI* umbrella, where the user actively attempts to control an application through purposeful thought [32]. Researchers often use motor tasks for this purpose, where finger tapping is most commonly used in fNIRS research [38, 156]. Research can also involve motor imagery [129], which consists of imagining a movement without actually performing it. Indeed, motor imagery has been shown to elicit similar brain activity to motor execution [112].

For *passive BCIs*, which are not used to voluntarily control an application, fNIRS data is used

to monitor and classify a user's mental state while they perform a task [32]. A range of mental workload tasks have been used to train such passive BCIs [10, 57, 66, 107]. One of the most popular is the n-back task, which involves remembering the recurrence of regularly presented stimuli [3, 95, 169]. A second task used to elicit mental workload is the word generation task, where subjects are asked to give as many words starting with a designated letter as possible [47, 70]. Finally, mental arithmetic tasks are often used, in which subjects are asked to solve simple mathematics operations such as additions, subtractions, multiplications, and divisions [70, 174]. Those different tasks have been described in more detail in 2.3.1.

Finally, it is worth mentioning that some researchers also looked at combining multiple modalities to build more reliable interfaces. For instance, work has been done to investigate hybrid fNIRS-EEG BCIs for mental workload classification [3, 146].

Use cases and applications

Examples of applications in terms of mental workload assessment are various. Such interfaces can be useful to evaluate HCI [56, 106, 158]. It can be used to evaluate how friendly or stressful a computer interface is. For example, Zijlstra et al., Bailey et al. or Cutrell et al. show how disruptive interruptions are when someone is performing a task [6, 40, 182]. We can use them to give a neurofeedback to subjects performing computer tasks. For example, Maior et al. built a BCI for real-time neurofeedback during computer-based tasks [108]. Those BCIs can be used to adapt tasks to the subject response, for example in a learning context with piano in their case [178]. Afergan et al. also used fNIRS to optimise workload during unmanned aerial vehicle path planning tasks [2].

Such interfaces are also useful to detect mental overload, especially in safety-critical environments. This is studied for example with flight crews by Ccakir et al. with fNIRS [27]. In a similar fashion, Laxmisan et al. have studied cognitive overload to prevent medical errors [94]. This can be also done for driver's safety since low or high mental workload leads to imperfect perception, insufficient attention, and inadequate information processing [21]. Therefore, cognitive load has been studied with fNIRS in studies about driving by Unni et al., Le et al., or Foy et al. for example [51, 95, 163]. Finally, passive BCIs can be used in neurorehabilitation to evaluate and adapt the quality of the therapy to the patient [93].

2.4 Machine learning with fNIRS

2.4.1 Machine learning

Machine learning is a subfield of artificial intelligence dedicated to computer algorithms that learn from data, without hardcoded rules [23, 59]. As defined by Mitchell [113]: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"*. The performance is usually measured with a loss function (error function) which can be for instance logistic loss (cross-entropy loss), hinge loss, or mean squared error.

The most common way of doing machine learning is with supervised learning. It consists in first collecting multiple input examples of data that are annotated with associated target outputs. The algorithm is then trained to predict the target outputs from the input examples alone. In machine learning, we call *example* a *"collection of features that have been quantitatively measured from some object or event that we want the machine learning system to process"* [59].

In the case of classification which is a type of machine learning task, the target outputs are discrete class labels, that the machine learning algorithm tries to predict: it will classify the examples. This is to be distinguished from other types of machine learning tasks such as regression for example in which the target outputs are continuous.

Many machine learning algorithms have been proposed in the literature, and some classification algorithms are very popular on a wide range of problems. For example:

- logistic regression can be used for classification [23]; it uses a sigmoid function with maximum likelihood estimation to output the probability of a predictor variable belonging to one of the classes;
- LDA, introduced by Fisher in 1936 [50], can be used as a classification algorithm by learning a linear decision surface to separate data into different classes [34];
- support vector machine (SVM), introduced by Bosser et al. in 1992 [18], can be used as a classifier by learning a hyperplane able to separate the data with a maximal margin with respect to data points of each class [64]; they can use a linear decision surface to separate the data (Figure 2.10), but also use kernel methods such as radial basis functions to learn a nonlinear function or decision boundary.

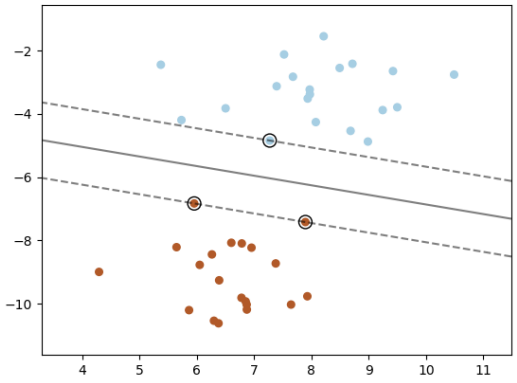

Figure 2.10: Example of linear SVM with two classes, showing the decision boundary and support vectors *(Source: scikit-learn.org)*.

2.4.2 Deep learning

Deep learning is a subset of machine learning based on ANN [59]. ANNs are inspired by the brain and are composed of neurons connected to each other with different weights [110]. One of the most basic neural network architectures is called the multilayer perceptron (Figure 2.11). It enables the transformation of an input, fed to the input layer, into output by modifying it in hidden layers thanks to the different weights of the neuron connections.

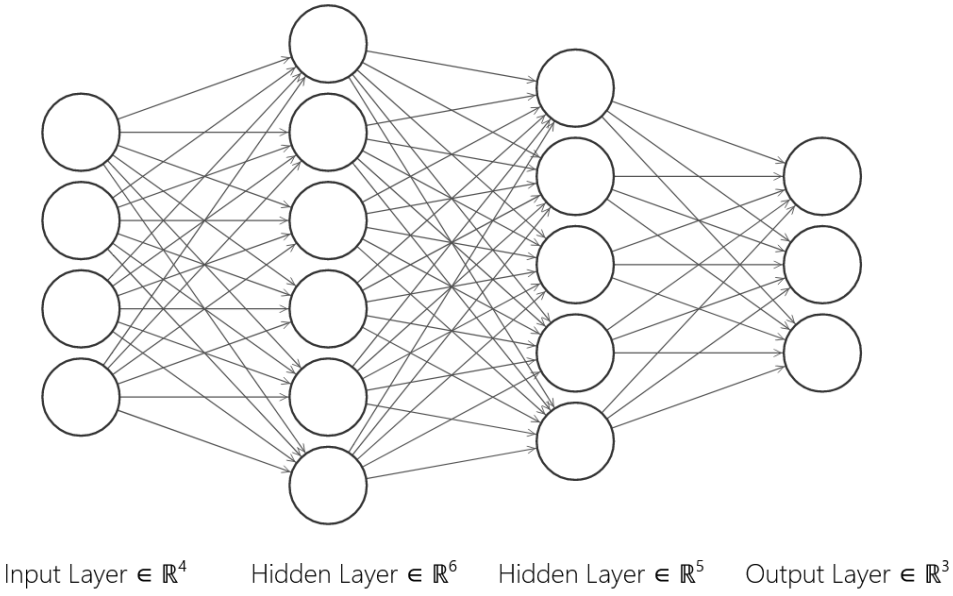

Figure 2.11: Simple multilayer perceptron artificial neural network.

More specifically, each neuron (also known as perceptron) is a multiple-input single-output parametric non-linear function. The output is defined as follows:

$$h = a(\sum_{i=1}^{n} w_i x_i + b) = a(w^\top x + b) \tag{2.11}$$

With:
- h the output;
- $x = [x_1, \ldots, x_i, \ldots, x_n]$ the inputs from the previous layer;
- $w = [w_1, \ldots, w_i, \ldots, w_n]$ the weights;
- b the bias;
- a the non-linear activation function.

One of the most common activation functions is sigmoid (Figure 2.12):

$$a(x) = \frac{1}{1 + exp(-x)} \tag{2.12}$$

Another one is the rectified linear unit (ReLU) activation function (Figure 2.12):

$$a(x) = max(0, a) \tag{2.13}$$

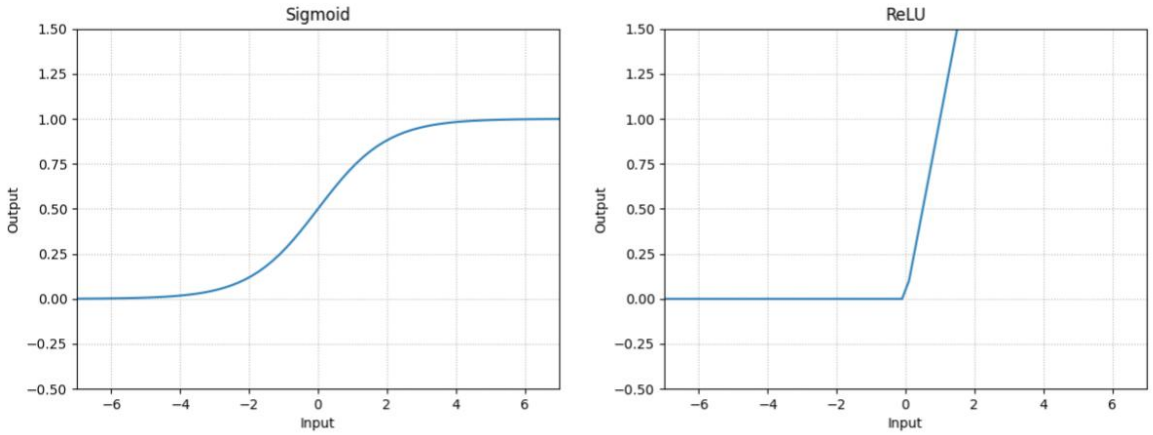

Figure 2.12: Activation functions (sigmoid on the left, rectified linear unit on the right)

In most cases, the weights are initialised randomly and are updated during the training of the neural network to better predict the output. More precisely, when training it, we use a set of inputs with known target outputs. Each input is fed one by one and the neural network predicts an output by transforming it thanks to the current values of its weights. The more different the predicted output is from the target output, the more the weights are modified in order to have a predicted output closer to the target. This modification of the weights is performed thanks to an algorithm called *backpropagation* which propagates the gradient of the error on the output backward to the neural network weights, from the output layer to the input layer all the way through the hidden layers. For training, the whole training data is usually fed multiple times into the model to update the weights of the model. An *epoch* consists in feeding the whole dataset to the model once, therefore a typical training will usually be performed on multiple epochs.

Building upon this concept of neural networks, more advanced models have been developed. We can describe for example the CNN [96], a neural network to which convolutional layers have been added. Those convolutions use kernels (filters) sliding along the axes of the input to transform it in a space-invariant way into a feature map, which enables the reduction of the input dimensionality. These types of models are very popular with images.

Another extension of neural networks is the recurrent neural network (RNN) [144]. They take advantage of recurrence to allow new inputs of a sequence to be treated in the context of previous inputs of that sequence: an input x_t at time t is fed into the neuron, outputting h_t, h_t being fed back into that neuron along with the input x_{t+1}, and so on. Such architecture is very popular with text, for example with a sequence of words: each word can be treated with the context of the previous word of the sequence.

2.4.3 Hyperparameters

As opposed to parameters that are internal to a model and are updated by the learning algorithm itself, hyperparameters are external to the model and rather control or influence the learning [12, 23, 59]. Each model will have its specific set of hyperparameters to control the learning, some models having none while others having more than 10. Hyperparameters can be discrete, this is the case for example of the type of kernel for SVMs or the choice of activation function for neural networks, while others are continuous, this is the case for example of the regularisation parameter for SVMs or the learning rate for neural networks. These hyperparameters can potentially have a high impact on how well a machine learning model will predict outputs [137]. They can be set manually or tuned by an algorithm.

2.4.4 Performance evaluation of machine learning models

Evaluating the performance of a machine learning model is evaluating its generalisation capabilities: how well does the model make predictions with data it has not seen during training?

With supervised learning, in practice when a dataset has been annotated, it is split into different parts [23, 59]. First of all, one part is held out for reporting the performance: this is the test set. It will only be used after the model has been trained and hyperparameters selected. The remaining data is split further in order to leave out another part: the validation set. This validation set is used for hyperparameter selection. The finally remaining data is then used for actually training the model: this is the training set. This process is described in Figure 2.13.

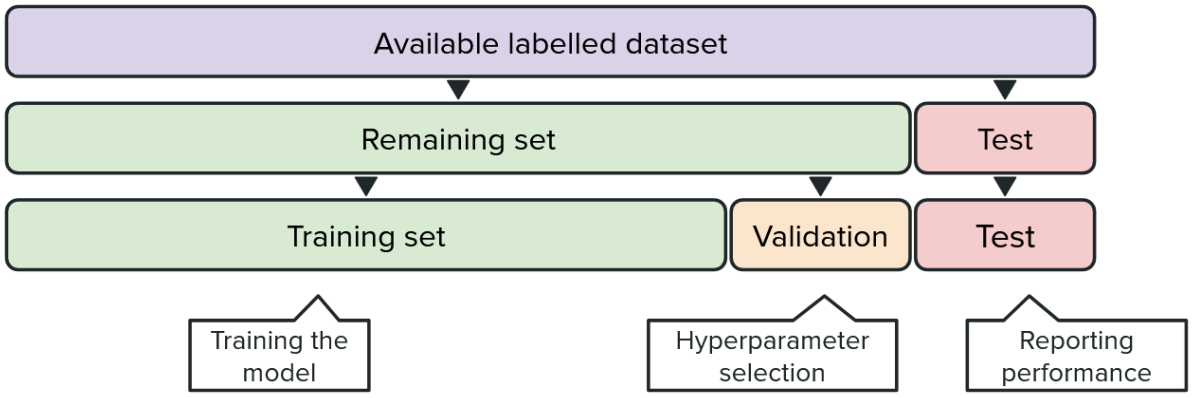

Figure 2.13: Training, validation, and test sets.

Performance can be assessed with different metrics. For classification, the most common ways of evaluating a model are with one or many of the following metrics:

- accuracy (number of correct predictions out of the total number of predictions);
- precision (number of correct predictions of a class out of the number of predictions of that class);
- recall (number of correct predictions of a class out of the number of expected predictions of that class);
- area under the receiver operating characteristic (ROC) curve (relation between recall and false positive rate).

Each of those metrics can be more or less useful depending on different factors such as the balance of classes or preferences on the type of error (eg. false positives are preferable compared to false negatives).

Hyperparameter selection is performed by training the model on the training set with different hyperparameter values that will be assessed on the validation set, the goal being to select the combination of hyperparameters yielding the best performance on the validation set. Multiple strategies for doing so are possible, some of the approaches are:

- *grid search* consists in testing exhaustively all combinations of specified hyperparameter values;
- *random search* instead tests a random set of combinations of hyperparameters;
- *Bayesian optimisation* consists in building a probability model to evaluate promising hyperparameter combinations.

Once the best combination has been selected, the model is then usually re-trained on the training and validation sets, to finally be evaluated on the test set that was not used until then.

This process is done because selecting hyperparameters is a process that requires knowledge of the outputs, therefore, if the hyperparameter selection was done on the test set intended for reporting the final performance, the results would not reflect the generalisation capabilities, namely the performance on unseen data. Indeed machine learning models can have a tendency to overfit, which means that they will predict very well data it has seen during training while predicting very poorly data they have never seen. We can see that this would not be very useful.

The splitting of the dataset can be done with different methods. A common one is to randomly split the data between training, validation, and test sets. This can be done in a stratified way, which means that the random split is made by preserving the percentage of samples for each class in each set as the initial distribution. However, in the case of small datasets, this technique is not optimal because the randomness can result in the extraction of a set poorly representative of the whole dataset. This is why in those cases cross-validation will be preferred, which consists in repeating the splitting multiple times with different subsets or splits of the original dataset. In practice, the most common approach to this is k-fold cross-validation in which the partition of the dataset is formed by splitting it into k non-overlapping subsets [59]. If this process is performed for leaving out both test and validation sets, it is called double cross-validation (nested cross-validation) [12]: using an outer loop cross-validation to evaluate the final performance on left out test sets and an inner loop cross-validation inside each outer loop split's remaining set in order to select hyper-parameters on left out validation sets for that outer split. This process is described in Figure 2.14.

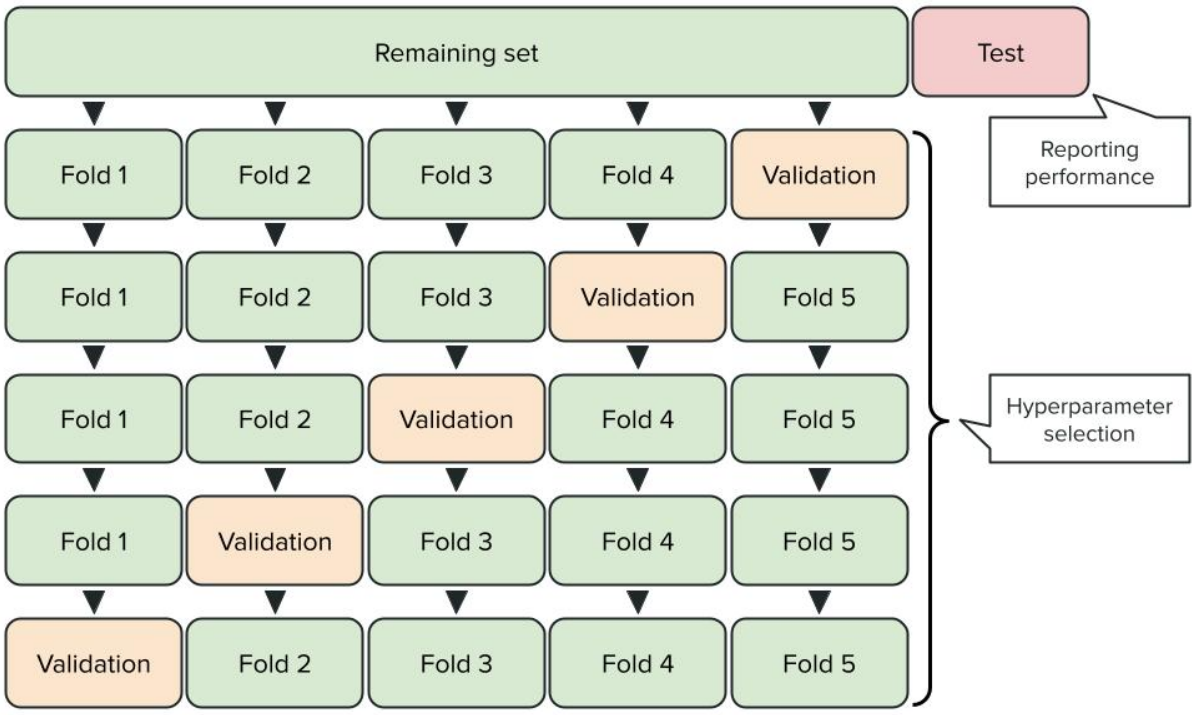

Figure 2.14: Example of inner 5-fold cross-validation.

2.4.5 Machine learning for fNIRS brain-computer interfaces

Ultimately, a BCI can be seen as a real-time classifier of brain data. In the vast majority of cases, this classifier is a supervised machine learning algorithm. Many traditional machine learning approaches have been used to classify fNIRS data in the context of different tasks and applications. For instance, Herff et al. [67] used LDA to classify mental tasks describing results of

71%, 70%, and 62% accuracy on mental arithmetic, word generation, and mental rotation tasks respectively against rest. Hong et al. [71] used LDA to classify between mental arithmetic, left- and right-hand motor imagery and obtained an average classification accuracy of 75.6%. Nazeer et al. [120] also used LDA with features extracted using vector-based phase analysis on finger tapping tasks presenting classification accuracies of 98.7% and 85.4% with 2 classes (left-hand, right-hand) and 3 classes (left-hand, right-hand, rest) respectively. Shin et al. [153] also used SVM for classifying mental arithmetic against baseline and obtained performances around 77% with eyes opened and around 75% with eyes closed. Other traditional machine learning models such as kNN have been also used, for example, to classify 3 different mental workload levels on n-back tasks and reached accuracies up to 52.08% [85].

Deep learning has also been used extensively with fNIRS data to classify activity during tasks [45]. For example, Chan et al. [30] used an ANN and reported an accuracy of 63.0% for the classification of mental signing against rest. Trakoolwilaiwan et al. [162] compared an ANN and a CNN to classify between left-, right-hand finger tapping and rest and report accuracies of 89.35% and 92.68% respectively. Yoo et al. [174] used a LSTM model for classification between mental arithmetic, mental counting, and puzzle solving, and reported accuracies up to 83.3%. Deep learning can be performed on raw data, which has the advantage of not relying on the subjectivity of feature extraction, contrary to some other traditional machine learning approaches that require it because of their difficulty in learning complex patterns.

The classification performances reported are extremely high and would suggest that this research is ready for technology transfer by industry, however, in some cases limitations in the methodology used for evaluation make that such results are an overestimation of the true performance on unseen data.

2.4.6 Machine learning crisis

While machine learning is a very useful tool, it can often be quite challenging to implement and evaluate models for researchers with limited experience. Furthermore, small mistakes in the implementation can lead to fundamentally flawed evaluations of the generalisation capabilities of models. This is amplified by the popularity and rapid development of machine learning, which often makes it hard to keep up to date with the latest methodology particularities, and results in potentially serious issues in published work. This has been shown by Kapoor and Narayanan [84] who did a meta-review of review papers pointing out flaws in applied machine learning to different scientific fields. They highlight the extent of machine learning issues including leakage, meaning that the evaluation data is not truly unseen data, and reproducibility issues, meaning that not enough details are available to replicate an existing work.

In order to address these issues they provide a taxonomy of issues divided in three main categories:

- missing training and test set separation (therefore potentially evaluating models on the training set);
- using illegitimate features (that should not be available when performing the generalisation task);
- having a test set not drawn from the distribution of interest (leading to poor ecological validity).

More specifically we can notice that leakage issues are found in various fields, including for example BCI research and neuroimaging in psychiatric research, in which many pitfalls can be found as reported in literature review papers [118, 170].

In addition to complexities related to machine learning itself, fNIRS data can also be challenging to fully comprehend. Indeed, there is still a lack of overall consensus in the fNIRS community

when it comes to collecting, processing, analysing, and reporting results. Some significant work however has been done in the past couple of years. Pinti et al. [134] did an extensive review of study design, processing, and statistical inference techniques used in the field and provided recommendations related to the method and reporting of fNIRS data pipelines. More recently, work by Yucel et al. [176] regrouping some of the most influential fNIRS researchers published a detailed paper about best practices for all things fNIRS, including the different types of fNIRS technologies, experiment design, pre-processing, signal processing, data analysis, modelling, and most importantly reporting, with a list of clear definitions for commonly used terms in fNIRS research. Such work currently has a significant impact on the improvement of standards for fNIRS research. However, the field is still lacking established best practices when it comes to machine learning methods applied to fNIRS. This can be highlighted by some apparent limitations of existing publications, as highlighted by a review done by Huang et al. [73].In an attempt to produce a benchmarking on a mental workload dataset, they noted the challenges of replicating existing work that could claim state-of-the-art performances.

All this helps us grasp the extent of issues around machine learning for fNIRS. This is the starting point for the subsequent research conducted in the context of this thesis and presented in the following chapters, in which we explore machine learning and its practices, more specifically for BCIs and mental workload assessment applications.

A journey of a thousand miles begins with a single step.

– Lao Tzu

Chapter 3

BenchNIRS: a benchmarking framework for fNIRS classification

- *RQ1*: How can we make the rigorous development, evaluation, and comparison of machine learning approaches for task classification from fNIRS simpler for researchers?

3.1 Motivations

Many research fields, including computer vision and natural language processing, benefit from strong standards, with state-of-the-art models, and established ways to benchmark machine learning on common datasets [90, 97, 105]. For relatively new areas of application, like the classification of fNIRS data, however, our community still lacks clear standards and approaches to compare and recognise significant advances in performance. Some tools are emerging in other related fields; for example, MOABB [79] enables the benchmarking of EEG-based BCIs, however, this software focuses on EEG paradigms which cannot be applied to fNIRS, and has limited support for implementing new machine learning (and especially deep learning) models.

The lack of machine learning standards in fNIRS more specifically creates a large discrepancy in how machine learning is applied to fNIRS, and how the methodology and results are reported in fNIRS papers, and this makes it hard to draw clear conclusions as to whether some approaches are really better than others. Notably, fNIRS machine learning papers sometimes fall foul of common mistakes, and the way that methods and results are presented is often missing critical information to make them reproducible. These problems are exacerbated by the fact that the field lacks commonly recognised open-access datasets for machine learning benchmarking, even though this journey is going in the right direction with the more frequent publication of open-access datasets and a will to gather them in a single place[1]. Moreover, the lack of code-sharing practices, which would enable inspection by others and improve reproducibility, is another issue that ultimately slows the progress of our field.

The same way the fNIRS community is going towards more established practices for signal processing [134, 147] and reporting [176], we aim in this research to provide a community resource specifically for machine learning in the context of fNIRS BCI applications. The work described in this chapter enables researchers to: 1) reuse the implementation of a robust machine learning framework methodology on common open-access fNIRS datasets in an open-source code repository; 2) share the implementation of fNIRS machine learning approaches such that they can be inspected and validated by others; 3) apply new machine learning approaches easily on multiple common open-access fNIRS datasets such that they can be compared to baseline implementations as well as recent contributions; 4) contribute to a community best practice checklist of expectations for both decisions made during implementation and analysis, and for reporting detail in papers.

[1] `https://openfnirs.org/data/`

Since in-depth comparisons of signal processing pipelines have already been conducted in the literature [134, 147], we will use as a starting point a signal processing pipeline based on those recommended best practices as a default pipeline and focus on the comparison of machine learning algorithms with a robust methodology. More specifically, we describe the implementation of a provided range of predefined baseline machine learning algorithms on a specific set of public datasets, which will later be used to present the results of such a multi-algorithm multi-dataset benchmarking comparison in Chapter 4. Finally, we present a recommendation checklist for researchers who are implementing machine learning approaches for classification from fNIRS data.

Further, we consider this work as a call to action, towards helping the community establish, from the variety of unstandardised approaches that have been published so far, consolidated best practices for identifying advances in our community. We list our initial recommended practices in this chapter, but we invite community members to contribute to a growing working document of best practices in a provided online repository[2].

3.1.1 Limitations of the current literature regarding machine learning for fNIRS

Recent work has produced strong examples for recommending best practices for processing fNIRS data [134], and considerations regarding the reporting of works with fNIRS [176]. In this trend, we highlight issues specific to machine learning classification with fNIRS data.

Machine learning approaches, while popular, often suffer from flaws in many existing publications across various domains of applications [84], including in the domain of BCIs [118]. In reviewing the literature of machine learning applied to fNIRS, it is common to see limitations that can be categorised into 2 types[3]. The first type regards the methodology, including potential mistakes, flaws, and lack of rigour. Common issues with published research include:

- not taking into consideration the experimental design when selecting instances to classify; this includes for example using resting periods for return to baseline in the same way as intentionally designed control baseline tasks, leading to a lack of ecological validity;
- randomly selecting hyperparameters without justification and not performing hyperparameter tuning, leading to a lack of robustness;
- optimising the model's hyperparameters using the test set (also called overfitting to the hyperparameters), leading to an overestimation of the performance;
- testing classifiers with data already seen during training and neglecting the potential overlap between the different sets (training, validation, test), this includes for example issues related to sliding windows with overlapping, leading also to an overestimation of performance;
- not using cross-validation or permutation testing to validate results, leading to a lack of robustness on small datasets;
- not performing a statistical analysis to compare results, leading to a lack of scientific validity;
- not handling class imbalance (for example as a result of the study design or the trial rejection), potentially leading to a misinterpretation of the results.

This first type of limitation is however difficult to highlight with certainty in most cases, this being related to the second type of limitation which is the reporting of works using machine learning with fNIRS. This makes the reproducibility of previous works often impossible, which is even more problematic when the data and/or the code are not available. Those limitations include:

[2] `https://gitlab.com/HanBnrd/benchnirs/-/blob/main/CHECKLIST.md`

[3] We choose not to call out specific papers and authors, but rather highlight things that researchers should look out for.

- not explaining what data is used as input of the classifier;
- not providing enough details regarding the machine learning models, including for example the hyperparameters or the architecture;
- not describing the split between training, validation, and test sets, and how many input examples are used (we call *example* an instance used as one input of a machine learning model [59]);
- not referring results to the number of classes or chance level.

All those issues often make it hard to be confident when critically evaluating machine learning papers with fNIRS, especially when it comes to reporting state-of-the-art results because they are not reproducible in most cases. Indeed, it may very well be the case that most of the variance across existing fNIRS machine learning publications is explained by methodology differences rather than actual model performance differences.

As a result, in this chapter, we will be interested in the following research question:

- *RQ1*: How can we make the rigorous development, evaluation, and comparison of machine learning approaches for task classification from fNIRS simpler for researchers?

In order to answer this question, we will here show how we created a tool to use a robust methodology for evaluating machine learning models for fNIRS in a reproducible way and presenting a true representation of performance on unseen data. That methodology relies on best practices from both the machine learning and fNIRS fields, and has been encapsulated into a Python-based framework called *BenchNIRS* with an application programming interface (API) enabling to implement easily all the steps to perform machine learning with fNIRS data on standard benchmarking open-access datasets, but also on any given fNIRS dataset. The following content of this chapter is a description of the framework's development, implementation, and usage.

3.2 *BenchNIRS* framework

BenchNIRS is a Python 3 open-source framework that has been designed for facilitating machine learning with fNIRS. For this purpose, it enables to load open-access fNIRS datasets, perform pre-processing, signal processing and feature extraction on fNIRS data, extract labelled segments of the data, and perform machine learning with training, optimisation, and evaluation of models.

3.2.1 Open-access fNIRS data loading

The framework was designed to support loading multiple open-access datasets by default. Indeed, *BenchNIRS* provides a function to load any of 5 open-access fNIRS datasets: n-back tasks from [66], n-back tasks from [155], word generation tasks from [155], mental arithmetic tasks from [154] and motor execution tasks from [7]. The focus was put here on mental workload tasks as it is an important domain of application for fNIRS as we saw in Chapter 2, but one dataset with a motor task was also used for comparison purposes. They were also chosen based on the characteristic of having at least a sampling frequency of 10 Hz as recommended by Yücel et al. [176] so that optode-scalp coupling can be assessed confidently (assuming a theoretical maximum heart rate of 220 pulsations per minute, so 3.67 Hz, it requires a minimum sampling frequency of 7.34 Hz as per the Nyquist-Shannon sampling theorem, rounded up to 10 Hz for good measure). All the datasets used are openly accessible and have been produced as part of previous studies by researchers of the fNIRS community. Appropriate ethical approvals were attained as stated in the datasets' dedicated papers, and participants gave written informed consent. Following up is a

description of the datasets based on their related publications, Table 3.1 summarises the size of each dataset and the different task conditions.

n-back task from Herff et al. 2014

This dataset consists of n-back tasks performed by 10 healthy participants. The experiment consisted, for each participant, of 10 epochs of each 1-back, 2-back, and 3-back; each epoch containing 3 ± 1 targets. Each epoch consisted of 5 seconds of instruction, 44 seconds of n-back with a letter every 2 seconds displayed for 500 ms, and a 15-second rest period. The data was recorded with an OxyMon Mark III from Artinis Medical Systems, with wavelengths of 765 and 856 nm and a sampling rate of 25 Hz. It is composed of 4 sources and 4 detectors on the PFC, resulting in 8 channels of HbO and 8 channels of HbR, with a source-detector distance of 35 mm. More details can be found in [66]. This dataset has been used for classification between 1-back, 2-back, and 3-back.

n-back task from Shin et al. 2018

This dataset consists of n-back tasks performed by 26 healthy participants. The experiment consisted, for each participant, of 9 epochs (divided into 3 sessions) of each 0-back, 2-back, and 3-back. Each epoch consisted of 2 seconds of instructions, 40 seconds of task, and 20 seconds of rest period. A random digit was given every 2 seconds displayed for 0.5 seconds and the targets appeared with a 30% chance. The data was recorded with a NIRScout from NIRx Medical Technologies, with wavelengths of 760 and 850 nm and a sampling rate of 10 Hz. It is composed of 16 sources and 16 detectors placed around the frontal, motor, parietal and occipital areas, resulting in 36 channels of HbO and 36 channels of HbR, with a source-detector distance of 30 mm. More details can be found in [155]. This dataset has been used for classification between 0-back, 2-back, and 3-back.

Word generation task from Shin et al. 2018

This dataset consists of word generation tasks performed by the same 26 healthy participants as the previous dataset. The experiment consisted, for each participant, of 30 epochs (divided into 3 sessions) of each word generation and baseline task. Each epoch consisted of a 2-second instruction showing an initial single letter for word generation or the fixation cross for baseline, a 10-second task period with a fixation cross, and a 13- to 15-second rest period also with a fixation cross. The hardware settings were the same as the previous dataset. More details can be found in [155]. This dataset has been used for classification between baseline task and word generation.

Mental arithmetic task from Shin et al. 2016

This dataset consists of mental arithmetic tasks performed by 29 healthy participants. The experiment consisted, for each participant, of 30 epochs (divided into 3 sessions) of each mental arithmetic and baseline task. Each epoch displayed the subtraction for 2 seconds, had a 10-second task period with a fixation cross, and a 15- to 17-second rest period also with a fixation cross. The data was recorded with a NIRScout from NIRx Medical Technologies, with a sampling rate of 10 Hz. It is composed of 14 sources and 16 detectors placed around the frontal, motor, and visual areas, resulting in 36 channels at 760 nm and 36 channels at 850 nm, with a source-detector distance of 30 mm. More details can be found in [154]. This dataset has been used for classification between baseline task and mental arithmetic.

Motor execution task from Bak et al. 2019

This dataset consists of finger and foot-tapping tasks performed by 30 healthy participants. The experiment consisted, for each participant, of 25 epochs of each right-hand finger tapping, left-hand finger tapping, and foot tapping. Each epoch contained a 2-second introduction, 10 seconds of actual task, and a 17- to 19-second rest period. The finger tapping was performed at 2 Hz and the foot tapping at 1 Hz. The data was recorded with a LIGHTNIRS from Shimadzu, with a sampling rate of 13.3 Hz. It is composed of 8 sources and 8 detectors around the motor cortex, resulting in 20 channels of HbO and 20 channels of HbR, with a source-detector distance of 30 mm. More details can be found in [7]. This dataset has been used for classification between right-hand finger tapping, left-hand finger tapping, and foot tapping.

Table 3.1: Information about the datasets including the different conditions, number of subjects and total number of blocs across all subjects, for each dataset. The total number of epochs is calculated as: *number of subjects* x *number of conditions* x *number of epochs per condition*.

Dataset	**Conditions**	***N* subjects**	**Total number of epochs**
Herff et al. 2014	1-back; 2-back; 3-back	10	300
Shin et al. 2018	0-back; 2-back; 3-back	26	702
Shin et al. 2018	baseline; word generation	26	1560
Shin et al. 2016	baseline; mental arithmetic	29	1740
Bak et al. 2019	right hand; left hand; foot	30	2250

3.2.2 Pre-processing of fNIRS data

In addition to loading fNIRS data from the open-access datasets, the software enables to pre-process it. This consists in converting the raw data measured by fNIRS (voltage of the measured light intensity data for each channel) into physiologically interpretable data, namely HbO and HbR concentration changes, using the modified Beer-Lambert law (MBLL).

Datasets from [66], [155], and [7] already provided HbO and HbR concentration change data while the dataset from [154] provided light intensity data. This is why data from [154] was automatically converted in *BenchNIRS* into optical density changes, relative to the average on the whole measurements for each channel. Then the MBLL [44] was applied to obtain changes in HbO and HbR. The [172] molar extinction coefficient table was used and the differential pathlength factor (DPF)s set to 6.0, as those are the most common in the literature and various fNIRS software. As per [154], the source-detector distances used were 3 cm.

This pre-processing relies on *NIRSimple*[4], a Python library developed in the context of this thesis and created for easily pre-processing fNIRS data, following the equations described in 2.1.1. It provides functions to convert light intensity fNIRS data or optical density fNIRS data into optical density changes relative to a reference or to the average across the whole measurement, and then to obtain concentration changes in hemoglobin (HbO and HbR), with a choice of different molar extinction coefficients datasets that can work for fNIRS brain recordings collected on adult or infants found in the literature [36, 139, 172, 183]. For each channel, the source-detector distance can be specified with different units, as well as the DPF and wavelengths that are used to match the extinction coefficients.

NIRSimple answers to the lack of a library to convert raw light intensity into concentration changes in Python. It has the advantage of working with any type of format as long as it can

[4] `https://github.com/HanBnrd/NIRSimple`

be opened with Python, and offers a wide choice of molar extinction coefficients, with all the most popular in the fNIRS community. It was important to develop such a tool in Python as this language becomes more and more popular thanks to its ease of use for machine learning, which is the focus of this thesis. Finally, *NIRSimple* integrates well with the MNE Python open-source library [60] which is one of the most popular for signal processing in neuroscience, with more than 200 contributors. *NIRSimple* is made available on PyPI[5] and its documentation can be found online[6] and in the appendices.

3.2.3 Signal processing of fNIRS data

Following up, *BenchNIRS* enables to perform signal processing on the fNIRS data with state-of-the-art methods. This is done relying on MNE-Python [61]. The HbO and HbR changes can be processed with TDDR [49] to correct motion artefacts. A bandpass, lowpass, or highpass infinite impulse response (IIR) Butterworth filter can also be applied, with adjustable order and cutoff frequencies. A bandpass filter of order 4 with cutoff frequencies of 0.01 and 0.5 Hz is recommended with the supported datasets to remove instrumental noise and noise related to heartbeat and slow drifts [119], without clashing with the experimental design of those datasets that can be loaded automatically with *BenchNIRS* (task durations ranging from 10 to 44 seconds resulting in task frequencies from $1/44 = 0.02$ Hz to $1/10 = 0.1$ Hz).

The original channels can be used, or they can be averaged by region of interest [119, 138] to end up with a left-hemisphere average and right-hemisphere average for each HbO and HbR in the appropriate brain area depending on the tasks of the supported datasets:

- mental workload tasks such as n-back, mental arithmetic, and word generation have been shown to elicit brain activity in the PFC so the hemisphere averages can be performed in that area for the mental workload task datasets [54, 119];
- motor execution has been shown to elicit brain activity in the motor cortex so the hemisphere averages can be performed in that area for the motor execution task dataset [14, 119].

This can be useful for having inputs of the same shape between datasets, as will be used in Chapter 4.

[5] `https://pypi.org/project/nirsimple/`
[6] `https://hanbnrd.github.io/NIRSimple/`

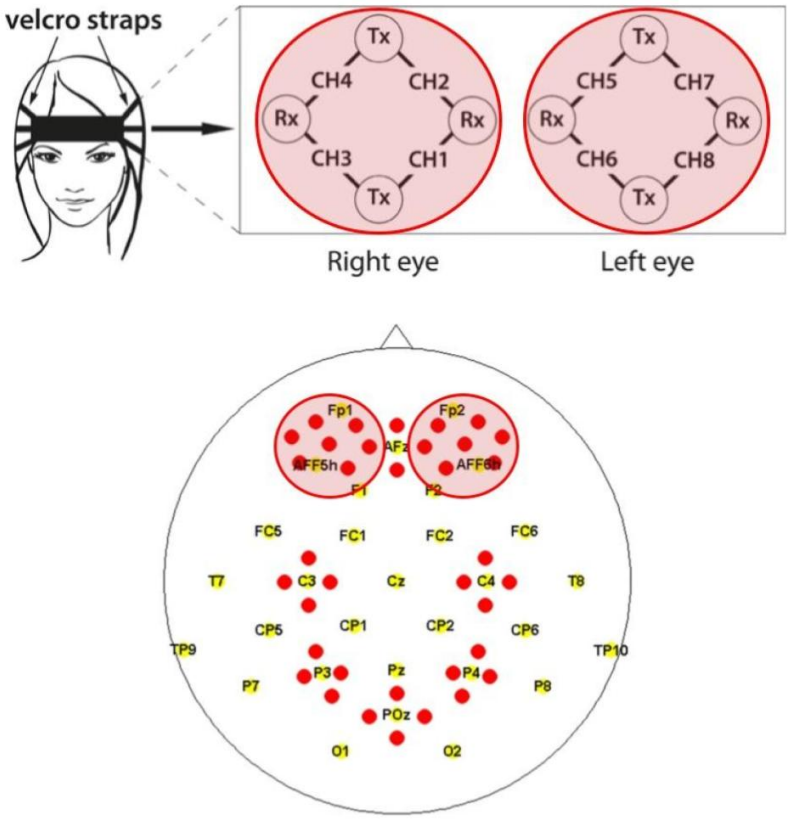

Figure 3.1: Maps of the regions of interest as circled in the red areas. *Top*: n-back dataset from Herff et al. 2014, fNIRS channels are represented by black lines with *CH* labels, each region of interest was an averaging of 4 channels. *Bottom*: n-back and word generation datasets from Shin et al. 2018, fNIRS channels are represented by red dots, each region of interest was an averaging of 7 channels.

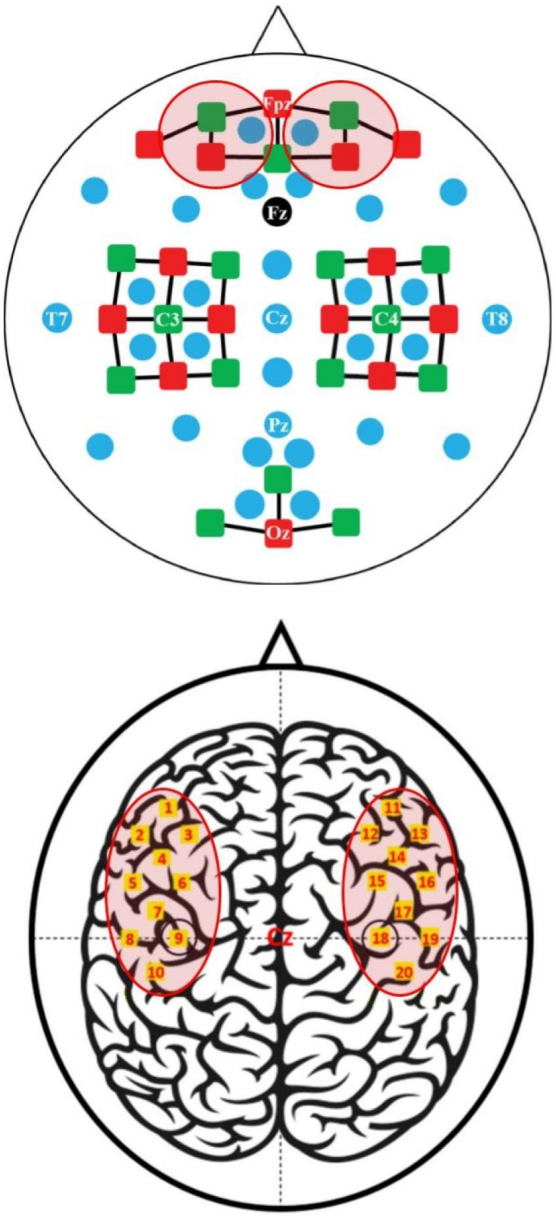

Figure 3.2: Maps of the regions of interest as circled in the red areas. *Top*: mental arithmetic dataset from Shin et al. 2016, fNIRS channels are represented by black lines between red (sources) and green (receptor) squares, each region of interest was an averaging of 4 channels. *Bottom*: motor execution dataset from Bak et al. 2019, fNIRS channels are represented by yellow squares, each region of interest was an averaging of 10 channels.

This results in a total of 4 regions of interest for each dataset. A detail of the regions of interest for each dataset can be found in Figures 3.1 and 3.2. This region of interest averaging is useful to have the same resulting number of channels for each dataset for the purpose of comparison and benchmarking between datasets, since different fNIRS devices with different number of optodes are used. Because this action was quite fastidious to achieve with MNE-Python initially, a contribution has been made to the library in order to simplify the API for combining channels by region of interest[7].

BenchNIRS then enables the epoching (extraction of segments of interest) of data based on

[7] `https://github.com/mne-tools/mne-python/pull/8014`

markers specific to each dataset (onset triggers and task duration), and a baseline correction can be performed such that the average on the baseline prior to each task is null for each channel or region of interest of each type. This baseline duration can be adjusted, though the duration as initially designed for each dataset is indicated in the documentation; this would typically be the duration of the instruction segments just prior to each task's onset trigger. The epochs are down-sampled to 10 Hz so that every dataset ends up with the same sampling frequency and to reduce computing demand for the following machine learning execution.

This signal processing relies on the open-source MNE-Python library [60] to return an MNE Epochs object[8] from which the data can be extracted as a matrix with three dimensions: the number of epochs, the number of channels, and the number of timepoints.

3.2.4 Data conditioning for machine learning

Following up, *BenchNIRS* provides a function to condition data for machine learning to fit a variety of approaches. This function crops out the baseline prior to the task used for correction, and enables to use epochs with full task duration, epochs with cropped task duration, or epochs split with a sliding window. The duration to which the epochs can be cropped can be adjusted, for example to select only the first seconds of the task. If used, the length of the sliding window for splitting the epochs into smaller segments can also be adjusted, which is useful to create more examples from a limited number of epochs. Finally, the data in M is converted into µM.

This function can be used not only with the supported datasets described in 3.2.1, but also with any dataset as long as the data is formatted as an MNE Epochs object[9]. It will in the end return an array with all the examples of conditioned fNIRS data, with the matching labels and the matching subject indices.

3.2.5 Feature extraction

Optionally, temporal features can be extracted with the API and used as input of machine learning models. The features that can be extracted here are 3 of the most popular in the fNIRS literature [119]:

- the mean of each channel or region of interest of each hemoglobin type across time for each example;
- the standard deviation of each channel or region of interest of each hemoglobin type across time for each example;
- the slope of the linear regression of each channel or region of interest of each hemoglobin type across time for each example.

3.2.6 Machine learning

The core of *BenchNIRS* is to provide functions to train, optimise and evaluate machine learning models for classification from fNIRS data.

Nested cross-validation is implemented [12]: evaluation is performed on the outer 5-fold cross-validation and the inner 3-fold cross-validation is used for hyperparameter selection (fine tuning) if any. A list of hyperparameters to be evaluated and selected with grid search can be provided. For each iteration of the outer cross-validation, after having selected hyperparameters, the model is re-trained with the best set of hyperparameters on the training and validation sets, to be evaluated on the test set; the maximum number of epochs for deep learning models can be specified and early stopping is performed using 20% randomly left out to avoid overfitting. This

[8] `https://mne.tools/stable/generated/mne.Epochs.html`
[9] `https://mne.tools/stable/generated/mne.Epochs.html`

consists in stopping deep learning training before the maximum number of epochs of if the loss on the 20% left out increased (non-strictly) for 5 consecutive epochs.

The data can optionally be normalised with min-max scaling, by computing minimums and maximums on the training and validation sets only (so excluding the test set for an unbiased normalisation) for each iteration of the outer cross-validation. The random state can be fixed so that the results are easily reproducible. The cross-validation splits are saved along with deep learning model weight, in order to later on load the trained models.

One can use this methodology with a subject-independent approach (unseen subject classification) where a group k-fold nested cross-validation will be performed, or a subject-specific approach where a stratified k-fold nested cross-validation (without shuffling to leave the test set out in order to address temporal leakage). The training set size can also be reduced to study the influence of its size.

Metrics including accuracy, precision, recall, and F1 score can be produced. Graphs are plotted and saved as images (with a colour-blind palette for accessibility) including training graphs (accuracy and loss), confusion matrices, as well as bar plots and graphs with 95% confidence intervals for overall result comparisons (examples are shown in Chapter 4).

While this machine learning methodology can be used easily with any deep learning model implemented using classes from the PyTorch library [126], the implementation of default models is also provided in *BenchNIRS* as benchmarking examples on the supported datasets, including LDA, SVC, kNN, ANN, CNN and LSTM. Following up is a description of those models.

Traditional machine learning

Firstly, 3 traditional machine learning models are implemented:

- LDA [34] classifiers learn a linear decision surface to split the data into different classes. They have the advantage of not having any hyperparameter to tune and a low computational cost. The LDA model implemented here uses features extracted from the signals as described in 3.2.5.

- SVM [64] classifiers or SVCs aim to find a hyperplane able to separate the data with maximal margin with respect to data points of each class. A SVC with a linear kernel or linear SVC uses a linear decision surface similarly to the LDA with the difference of it being fitted with margin maximisation. It uses a regularisation hyperparameter that needs to be tuned. The linear SVC implemented here also uses the features extracted from the signals. The regularisation parameter can be optimised following the hyperparameter selection procedure described previously. The maximum number of iterations is set to 250,000 in order to guarantee convergence.

- kNN [4] is a non-parametric classification algorithm using the closest k points from the training data in order to make a prediction on the class. Here, we use a majority vote with those labelled k points. The kNN classifier implemented here also uses features extracted from the signals, and the algorithm uses a uniform weighting of the k nearest neighbours meaning that each point is weighted equally in the voting. The number of neighbours k is a hyperparameter of the algorithm that can be tuned according to the same procedure as the other models described previously.

BenchNIRS relies on Scikit-learn [128] for the implementation of those traditional machine learning models.

Deep learning

Secondly, 3 deep-learning models are implemented:

- ANNs [110] are the simplest type of neural network. Neural networks are composed of units called artificial neurons arranged into layers, whose outputs are computed by a non-linear function of the weighted sum of its inputs from the previous layer. This process happens until the last layer, where a probability distribution for the classes is computed, enabling to get a prediction. The ANN implemented here uses temporal features extracted from the signals as well, and works on data with region of interest averaging as described in 3.2.3. Hence, the input layer is composed of 12 neurons, followed by 2 fully connected layers of respectively 8 and 4 neurons, finished by an output layer of 2 or 3 neurons depending on the number of classes for the dataset. It uses rectified linear unit (ReLU) as the activation functions for the hidden layers and a cross-entropy loss. The Adam optimiser was used due to its good and reliable performance across many deep learning problems [149]. It is an optimiser based on adaptive estimates of lower-order moments [88]. The learning rate and mini-batch size can be optimised following the hyperparameter selection procedure described previously.

- CNNs [96] are an extension of neural networks to which convolutional and pooling layers have been added. They use kernels sliding along the input dimensions so that it is transformed in a space-invariant way, in an attempt to simplify complex patterns by learning how to extract features. A CNN is typically composed of convolutional layers followed by standard neural network layers. The CNN implemented here works on 100-timepoint inputs (10 seconds of 10 Hz data) with region of interest averaging, and therefore does not use any temporal feature extraction. It is composed of 2 one-dimensional convolutional layers across the time axis: the first one with 4 input channels and 4 output channels, a kernel size of 10 (one-dimension kernel), and a stride of 2; the second one with 4 input channels and 4 output channels, a kernel size of 5 (one-dimension kernel) and a stride of 2. Each convolutional layer is followed by a one-dimensional max pooling across the time axis with a kernel size of 2. Those convolutions and max poolings are followed by 2 fully connected layers of 20 and 10 neurons respectively, followed to finish with by an output layer of 2 or 3 neurons depending on the number of classes for the dataset. It uses rectified linear unit (ReLU) as the activation functions for the hidden layers, Adam as the optimiser, and a cross-entropy loss. The learning rate and mini-batch size can be optimised following the procedure described previously.

- LSTM neural networks [69] belong to the family of RNNs. RNNs [144] can be seen as simple neural networks allowing new inputs of a sequence to be treated in the context of previous inputs of that sequence. LSTMs are an extension of that using a memory cell in order to learn longer-term dependencies. They are useful compared to RNNs as they allow to overcome the vanishing gradient problem. The LSTM implemented here uses the signal processed data without feature extraction as well, also with region of interest averaging. It uses one LSTM recurring unit with an input size of 80 (each input being arranged as a sequence of elements of 20 timepoints (2 seconds of 10 Hz data) on 4 regions of interest) and a hidden size of 36. It is then followed by a fully connected layer of 16 neurons, followed to finish with by an output layer of 2 or 3 neurons depending on the number of classes of the dataset. The model uses rectified linear unit (ReLU) as the activation functions for each hidden layer except the LSTM unit using hyperbolic tangent (tanh), Adam as the optimiser, and a cross-entropy loss function. Again, the learning rate and the mini-batch size can be optimised following the same procedure described previously.

BenchNIRS relies on PyTorch [126] for the implementation of those deep learning models.

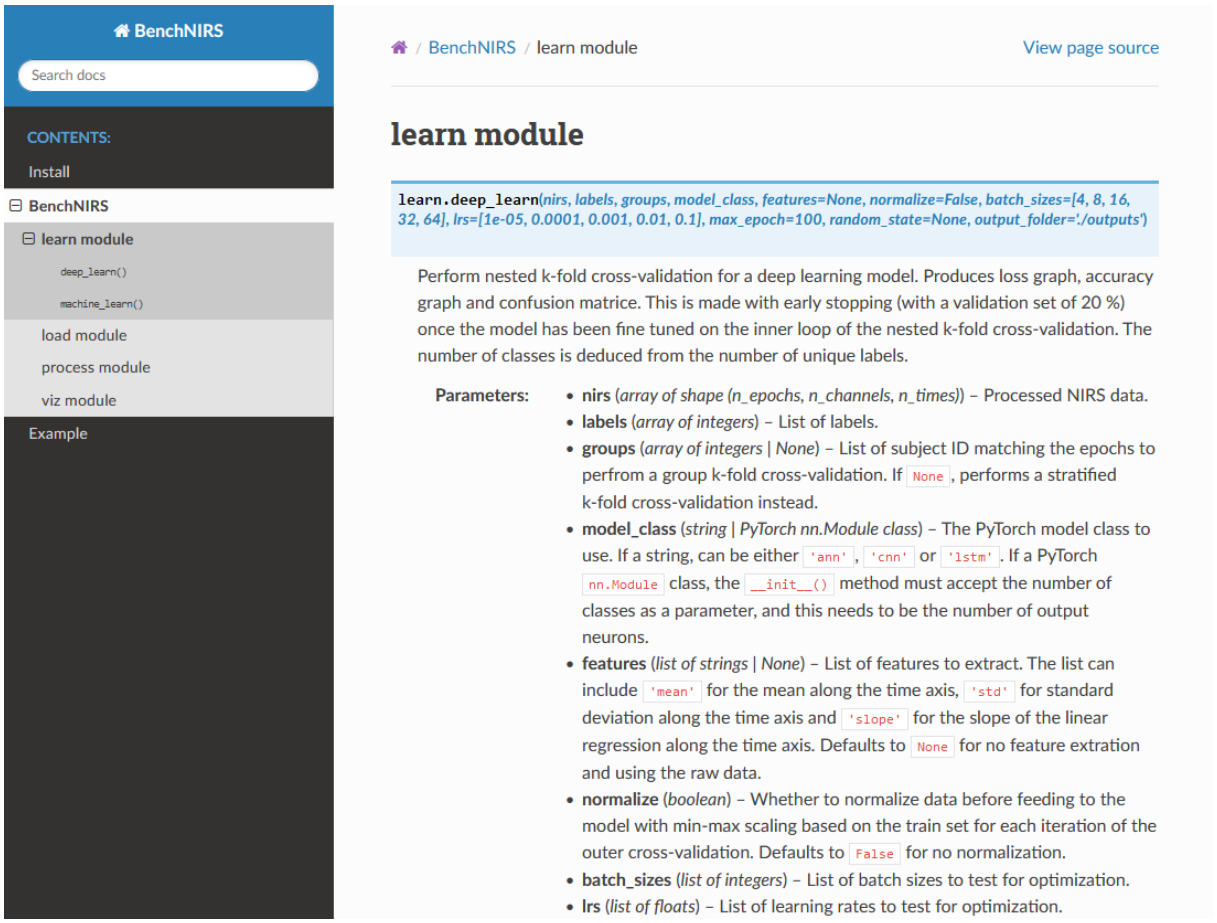

Figure 3.3: Screenshot of the online documentation of a *BenchNIRS* function.

3.2.7 Documentation, installation and use case

Docstring (special text used in source code to provide documentation) is present in the code which enables support to have documentation right within integrated development environments. The framework also benefits from an extended documentation that can be accessed online[10]. A screenshot of that documentation is presented in Figure 3.3. All the functions described previously are detailed along with their different parameters and outputs. This extended documentation also has installation instructions (with requirements), a recommendation checklist (detailed in 3.3.1) and use case examples demonstrating for instance how the framework can be used to train, fine-tune and evaluate a CNN model for task classification on the 5 supported datasets. The documentation of *BenchNIRS* can be found in the appendices of this thesis.

Instructions on how to install the framework and download the open-access datasets supported by it are also found in the documentation. The full framework including examples and recommendations can be downloaded from GitLab, or the package containing the core functions described previously can be directly installed using *pip* with a simple command:

To sum up, we use two different terms to define *BenchNIRS*: *framework* and *software package*. The *software package* contains the Python 3 functions (with their docstring) for loading the datasets, performing pre-processing, signal processing, feature extraction, and machine learning and can be downloaded and installed with all the requirements from PyPI using *pip*[11] with the following command:

```
pip install benchnirs
```

[10] `https://hanbnrd.gitlab.io/benchnirs/`
[11] `https://pypi.org/project/benchnirs/`

Meanwhile, the *framework* includes the *software package* but also the extended documentation, example scripts, and the recommendation checklist for machine learning with fNIRS. It also contains some examples of statistical analysis to compare results to chance level and to compare models to each other. It is in the form of an online Git repository and can be downloaded from GitLab[12].

In practice, *BenchNIRS* can be used in different use cases:

- It can be used when a user wants to use a default provided model with an open-access dataset supported by the framework, for example to reproduce benchmarks. This can be done in a couple of lines of Python 3 code:

```
import benchnirs as bn
epochs = bn.load_dataset('shin_2018_nb')
data = bn.process_epochs(epochs['0-back', '2-back', '3-back'])
results = bn.deep_learn(*data, 'ann')
print(results)
```

- However, *BenchNIRS* is mainly provided as a tool to facilitate the development of new machine learning models. As such, users can develop their own deep learning model classes with PyTorch, and use the framework to train those models for classification on any of the 5 supported datasets. An example of such practice, after defining a PyTorch model class called `my_model` would be the following lines of code:

```
import benchnirs as bn
epochs = bn.load_dataset('herff_2014_nb')
data = bn.process_epochs(epochs['1-back', '2-back', '3-back'])
results = bn.deep_learn(*data, my_model)
print(results)
```

- Finally, the framework can also be used to develop new machine learning models on other datasets, as long as they can be loaded as MNE Epochs objects. This can be done (assuming the user defined a PyTorch model class `my_model` and a dataset Epochs variable `my_epochs`) with the following lines of code:

```
import benchnirs as bn
data = bn.process_epochs(my_epochs)
results = bn.deep_learn(*data, my_model)
print(results)
```

3.2.8 Development

BenchNIRS was designed as a collaborative Git repository right from the beginning. It is available in open source on GitLab[13] under the GNU General Public License v3+. Improvements were made iteratively following a Git workflow for continuous delivery, where the main branch of the project is always in a working state, and improvements are made through feature branches that are merged to the main branch after ensuring everything still works as expected. Before merging, each of the improvements is detailed in a changelog to inform the users of changes between versions. The development follows Python's PEP 8 conventions to produce a high-quality and easily maintainable code, documented following the NumPy Python Style Docstring[14]. Automatic pipelines facilitate continuous integration and continuous delivery by automatically building the online documentation using Sphinx[15] and automatically publishing the software package on the PyPI[16] software repository for easy installation by the users using *pip*.

[12] `https://gitlab.com/HanBnrd/benchnirs`
[13] `https://gitlab.com/HanBnrd/benchnirs`
[14] `https://numpydoc.readthedocs.io/en/latest/format.html`
[15] `https://www.sphinx-doc.org/en/master/`
[16] `https://pypi.org/project/benchnirs/`

3.3 Discussion

3.3.1 Best practices for machine learning with fNIRS

The development of *BenchNIRS* made us reflect on best practices for machine learning specifically applied to fNIRS, which resulted in the proposition of a set of recommendations.

First and foremost, we would like to encourage fNIRS machine learning researchers to follow fNIRS specific guidelines already described in important previous work for signal processing with [134] and [147], but also best practices for publications with [176]. Further to these, to complement our answer to *RQ1* and in line with other fields of application [115], we provide recommendations that we believe important when using machine learning for classification from fNIRS data, based upon the practice of making our framework. Some of these recommendations have become standard processes in machine learning, but aspects are often missed in recent machine learning papers within the fNIRS community.

The first recommendations are methodology-related which we list as the following checklist:

- Is the classification goal adequate with regards to the experimental design of the dataset (for example to avoid using return to baseline as control baseline epochs or to avoid bias related to order effect)?
- Has nested cross-validation (also called double cross-validation) been used, with the outer cross-validation (leaving out the test sets) for evaluation and the inner cross-validation (leaving out the validation sets) for the optimisation of models?
- Have the hyperparameters been optimised on validation sets and with what method (eg. grid search, random search, etc.)?
- Have test sets been used for evaluation and nothing else (no optimisation should be performed with the test set)?
- Have the training, validation, and test sets been created in accordance with what the model is hypothesised to generalise (eg. unseen subject, unseen session, etc.), using group k-fold cross-validation if necessary?
- Has it been ensured that test data was not included when performing normalisation?
- Has it been ensured that there was no overlap between training, validation, and test sets, especially when using a sliding window for the extraction of epochs?
- If in the presence of class imbalance, has it been taken into account with the use of appropriate metrics (eg. F1 score)?
- Has a statistical analysis been performed to demonstrate the significance of the results compared to chance level and other models?

Where relevant, these points have been implemented in our framework and we therefore encourage researchers to use it for time-saving and reproducibility purposes.

The second recommendations are related to the reporting which we list as the following checklist:

- Has the data used as input for the classifier and its shape been described?
- Has the number of input examples in the dataset been stated?
- Have the details related to the cross-validation implementation and whether it involves data shuffling been provided?

- Have the details of each model used including the architecture of the model and every hyperparameter been provided?
- Has it been described which hyperparameters have been optimised and how?
- Have the number of classes and chance level clearly been stated?
- Have all the information related to the statistical analysis of the results, including the name of the tests, the verification of their assumptions and the p-values, been provided?

Finally, we invite researchers to have a look at guidance from the machine learning community regarding reproducibility[17] and research code publication[18].

We make this list of recommendations available in the framework as a checklist[19], and hope this can act as a starting point for the community to contribute to a more exhaustive checklist for the field of machine learning applied to fNIRS.

3.3.2 Community contributions

To give concrete material to answer *RQ1*, the framework developed here, including the list of recommendations from machine learning with fNIRS is made available as an online Git repository[20].

The repository is open to community contribution in order to improve the recommendation checklist, add support for new open-access datasets, improve the implementation of the machine learning methodology and add support for new machine learning approaches, or improve the production of results and figures. Guidance on how to contribute can be found on the repository page.

Furthermore, we encourage researchers to use the framework if they wish to compare the results obtained with their machine learning models on the datasets supported by *BenchNIRS* with the proposed methodology.

3.4 Summary

This work has introduced an open-source framework called *BenchNIRS* for machine learning with fNIRS. It enables to train, optimise and evaluate machine learning models for classification from fNIRS data, and more specifically to perform the benchmarking of machine learning with fNIRS enabling researchers to robustly validate classification results on 5 open-access datasets published by the community.

We invite the fNIRS community to use the framework when performing classification with their own machine learning models for a convenient evaluation on open-access data. We welcome contributions to extend and strengthen the guidelines that we propose as well as the implementation of the machine learning methodology.

[17] `https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf`
[18] `https://github.com/paperswithcode/releasing-research-code`
[19] `https://gitlab.com/HanBnrd/benchnirs/-/blob/main/CHECKLIST.md`
[20] `https://gitlab.com/HanBnrd/benchnirs`

Chapter 4

Benchmarks and influencing factors for fNIRS classification

- *RQ2*: What are the benchmarks of popular machine learning models on various tasks from open access fNIRS datasets?
- *RQ3*: Across these benchmarks, what factors influence the machine learning classification accuracy?
  - *RQ3a*: What is the influence of the number of examples used for training machine learning models on classification accuracy?
  - *RQ3b*: What is the influence of the time window length of inputs on classification accuracy?
  - *RQ3c*: What is the influence of using a sliding window on classification accuracy as opposed to using epochs starting at the beginning of each task?
  - *RQ3d*: What is the influence of using a subject-specific approach (classification of data within subjects) on classification accuracy as opposed to a subject-independent approach (classification of data from unseen subjects)?

4.1 Introduction

Directly following up on the work done in Chapter 3, we here take *BenchNIRS* in a concrete use case, to provide along with it a benchmarking of popular machine learning models for the classification of fNIRS data on 5 open-access datasets.

We therefore ask the following research question:

- *RQ2*: What are the benchmarks of popular machine learning models on various tasks from open access fNIRS datasets?

However, this question cannot be answered by itself without considering the huge variety of approaches and settings when performing machine learning with fNIRS.

First, the datasets used by different researchers can vary greatly in size, and consequently in the number of examples for each of the classes. On the OpenfNIRS website[1], we can for example find open-access datasets with fNIRS recordings ranging from 7 to 43 subjects [58, 177]. It is argued that the dataset size can influence the performance of machine learning and more specifically deep learning models [59], so we can ask ourselves what is the actual impact of this dataset size in the context of fNIRS data.

[1] `https://openfnirs.org/data/`

Also, some authors will prefer a subject-specific approach while others prefer a subject-independent approach. For example, Hennrich et al. [65] or Trakoolwilaiwan et al. [162] used subject-specific approaches, while Lim et al. [98] or Liu et al. [101] investigated subject-independent approaches.

Furthermore, the duration of data segments used as inputs of models and the way they are extracted can also vary depending on the task, study design decisions, and constraints. For example, Herff et al. studied window durations ranging from 5 to 40 seconds [66], while Zafar et al. used different window durations spanning from 1 to 5 seconds at different placements after the onset trigger [179], and Shin et al. used a 3-second sliding window[153].

This is why an important point to complement benchmarking is to study the influence of those different factors on the classification performance.

Therefore, we ask the subsequent research question:

- *RQ3*: Across these benchmarks, what factors influence the machine learning classification accuracy?

More specifically, as we delve into the specifics of what influences classification accuracy, we ask:

- *RQ3a*: What is the influence of the number of examples used for training machine learning models on classification accuracy?
- *RQ3b*: What is the influence of the time window length of inputs on classification accuracy?
- *RQ3c*: What is the influence of using a sliding window on classification accuracy as opposed to using epochs starting at the beginning of each task?
- *RQ3d*: What is the influence of using a subject-specific approach (classification of data within subjects) on classification accuracy as opposed to a subject-independent approach (classification of data from unseen subjects)?

In this chapter, we will then present the methodology and results of all those benchmarks, made using the *BenchNIRS* framework.

4.2 Methods

We evaluated 6 machine learning models (LDA, SVC, kNN, ANN, CNN and LSTM) on the 5 open-access datasets supported *BenchNIRS* (two n-back datasets, a word generation dataset, a mental arithmetic dataset, and a motor execution dataset) in a multi-model multi-dataset benchmarking. Each dataset was analysed separately, and statistical tests were performed (using SciPy [166]) to answer the different research questions stated in the introduction, with a threshold of significance set at 5% for all of them. All the methods described below have been implemented, relying on *BenchNIRS*, and are made available in open source along with the framework on GitLab[2].

4.2.1 Data

The 5 open-access fNIRS datasets that can be loaded with *BenchNIRS* were used for this multi-dataset multi-model comparison, for which more information can be found in Chapter 3.

The first one is the dataset of n-back task collected by Herff et al. [66], containing 3 classes: 1-, 2-, and 3-back. It was composed of recordings of 10 subjects for a total of 100 examples per class.

[2] `https://gitlab.com/HanBnrd/benchnirs`

The second dataset is from Shin et al. who recorded also n-back tasks [155], with 3 classes: 0-, 2-, and 3-back. It was recorded on 26 subjects for a total of 234 examples per class.

The third is word generation data collected by Shin et al. during the same study as the n-back one [155]. It was composed of data of 2 classes: word generation, and baseline; for a total of 780 examples per class recorded on the same 26 subjects as the n-back study.

The fourth dataset collected by Shin et al. was composed of data from a mental arithmetic task [154], composed of 2 classes: mental arithmetic, and baseline. Recordings from 29 subjects resulted in a total of 870 examples per class.

Finally, the fifth dataset of motor execution tasks was collected by Bak et al. [7], composed of 3 classes: right hand, left hand, and foot tapping. This was recorded on 30 subjects with a total of 750 examples per class.

4.2.2 Signal processing and feature extraction

All those datasets were processed with the same pipeline using *BenchNIRS*.

The data was first converted into HbO and HbR for the dataset that only provided light intensity data, namely the dataset of mental arithmetic tasks from Shin et al. [153]. This was done by converting into optical density changes relative to the average on the whole measurements for each channel, and then applying the MBLL [44] with the molar extinction coefficient table from [172] and a DPF of 6.0.

TDDR [49] was then applied on all the datasets to remove motion artefacts, followed by a Butterworth IIR bandpass filtering of fourth order with cutoff frequencies of 0.01 and 0.5 Hz.

Then, the channels were averaged by regions of interest, one per hemisphere, for the purpose of comparison (to have the same number of channels for each dataset) following Figures 3.1 and 3.2 as described in Chapter 3, and the data was downsampled to 10 Hz for every dataset.

Epoching was performed using the task onset triggers and with a baseline correction using 2 seconds prior to the task onset, corresponding to the shortest duration of instruction amongst the datasets from [155] and [7] for a fair comparison, meaning that longer durations would be cropped down to 2 seconds.

Finally, every epoch of every dataset was cropped down to the shortest epoch duration available of 10 seconds from the onset triggers ([155], [154] and [7]), so that examples would have the same shape across all the datasets.

In the end, the shape of each example was 4x100, representing HbO and HbR in each hemisphere multiplied by 100 timepoints (10 seconds of 10 Hz signals). This is what was used as input for the CNN and LSTM models.

Temporal features have been extracted and used as input for 4 of the 6 models tested here: the LDA, SVM, kNN, and ANN. This was not done for the CNN to let it extract features from raw data and the LSTM to let it learn temporal dependencies from the raw data. The features extracted here are 3 of the most popular in the fNIRS literature [119]:

- the mean for each region of interest of each type across time;
- the standard deviation for each region of interest of each type across time;
- the slope of the linear regression for each region of interest of each type across time.

So for the LDA, SVC, kNN and ANN models, one input consisted of the mean, standard deviation and slope of the linear regression for each hemisphere and each chromophore simultaneously, resulting in a shape of 4x3 per example.

4.2.3 Machine learning

6 supervised machine learning models were compared on all the datasets, this includes 3 traditional machine learning models and 3 deep learning models. Those models were the default

models provided along with the framework and more details about them can be found in Chapter 3. They are summarised here thereafter.

For the traditional machine learning, the following models were evaluated:

- LDA
- SVC with linear kernel
- kNN with uniform weighting

For deep learning, the following models, described in Chapter 3 with more details, were evaluated:

- ANN
- CNN
- LSTM

Nested cross-validation was used as implemented in the framework for hyperparameter selection and evaluation of the models.

The hyperparameters optimised for the traditional machine learning approaches were the regularisation parameter for the SVC and k the number of nearest neighbours for the kNN. For the deep learning approaches, the learning rate and the mini-batch size were optimised, as those parameters are known to influence models the most [12]. The optimisation was done within the following ranges using grid search following common machine learning practice [12, 128]:

- the regularisation parameter's values tested were 0.001, 0.01, 0.1 and 1;
- the values of k (number of nearest neighbours) tested were the integers from 1 to 9;
- the learning rate's values tested were 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1};
- the mini-batch sizes tested were 4, 8, 16, 32 and 64.

All those models were trained on the CPU (Intel Xeon E5 v3 processor) as GPUs did not provide much time advantage in this case (small dataset sizes, time series data).

4.2.4 Subject-independent approach

The first approach evaluated is the subject-independent approach, in which the generalisation goal consists in classifying the task's condition (class) from fNIRS data on unseen subjects.

For the nested cross-validation, the outer cross-validation consisted of a group 5-fold cross-validation to leave the test set out, such that the same subject's data cannot end up in both the training and validation set and the test set. This outer cross-validation was used to evaluate the different machine learning models for each dataset. This was made so that the results reflect the performance of a classifier of data from unseen subjects. The inner cross-validation consisted of another group 3-fold cross-validation to separate training and validation sets. This inner cross-validation was used for hyperparameter selection, choosing hyperparameters based on the accuracy on the validation set (accuracy was chosen since the classes of every dataset were perfectly balanced).

In addition to that, early stopping was performed for the deep learning models to avoid overfitting. This early stopping was done after the best hyperparameters were selected and the model was retrained on the whole training and validation sets except 20% randomly left out to perform this early stopping. This consisted in stopping deep learning training before the maximum number of epochs of 100 if the loss on the 20% left out increased (non-strictly) for 5 consecutive epochs.

Overall, the parts of the method that could be affected by the random seed were:

- the shuffling of the training and validation set (this has an influence on the models with hyperparameter optimisation);
- the selection of the 20% validation set used for early stopping;
- the weight initialisation of the deep learning models.

Regarding the results, the accuracies on each of the 5 outer folds were averaged to determine the overall accuracy for each model of each dataset. This metric was used for reference rather than others such as F1 score because of its simplicity and the perfect class balance of each dataset (no epoch rejection was performed).

For each model and each dataset, a one-tailed t-test was used (using the accuracy values on each outer fold) to determine whether its accuracy was greater than chance level if the distribution of the accuracies of the model on the outer folds followed a normal distribution as tested with a Shapiro test; otherwise, a one-tailed Wilcoxon test was used.

Next, a statistical analysis was run with the accuracy values on each outer fold to compare models to each other within each dataset. For this purpose a one-way analysis of variance (ANOVA) test was performed if the normality and the homoscedasticity were not excluded with Shapiro tests and a Bartlett test respectively, otherwise, a non-parametric Kruskal–Wallis test was used. If an effect of the model was identified, one-tailed paired t-tests with Bonferroni correction were used to compare each model against the others.

In addition to those results, a confusion matrix was produced for each model of each dataset, comparing the predictions made across all the outer folds against the true class labels (those can be found in the appendices).

4.2.5 Influence of the number of training examples (subject-independent approach)

In addition to comparing the different models, the influence of the training set size was also studied with this subject-independent approach. For that purpose, after leaving out the test set, a proportion of the training and validation set was discarded using the *BenchNIRS* API. This way we studied variations from 0 to 50% discarded of the training data by stride of 10% for every dataset. The same procedure was then applied in terms of validation and hyperparameter search.

The correlation between the training set proportion used and the accuracy was studied with a Pearson's correlation test for each model of each dataset if the assumption of normality was verified as per a Shapiro test, otherwise, a Spearman test was used.

4.2.6 Influence of the time window length (subject-independent approach)

The influence of the time window length was also studied with the subject-independent approach. We compared here epochs of 2, 3, 4, 5, 6, 7, 8, 9, and 10 seconds from the onset trigger marking the start of each task. Again, the same procedure of validation and hyperparameter search was used.

The same way as before, correlations between the window length and the accuracy were studied with a Pearson's correlation test for each model of each dataset if the assumption of normality was verified as per a Shapiro test, otherwise a Spearman's correlation test was used.

This approach enabled the comparison of the 4 models using feature extraction: LDA, SVC, kNN, and ANN. This is because comparing the models using the data without feature extraction would have required changing the architecture of those models for each window length which would have added too many variables in the comparison.

4.2.7 Subject-independent approach with sliding window

Furthermore, the same procedure as the subject-independent approach was used but with a 2-second sliding window on the 10-second epochs instead of the 10 seconds at once. No overlapping was used between the different time windows in this case, so it was equivalent to splitting each 10-second epoch into five smaller 2-second epochs, multiplying the total number of epochs by five. Therefore, it enabled us to build models making each inference from only 2 seconds worth of fNIRS signal.

This approach enabled the comparison of the 4 models using feature extraction: LDA, SVC, kNN, and ANN. This is because comparing the models using the data without feature extraction would have required changing the architecture of those models compared to the initial approach.

4.2.8 Subject-specific approach

Finally, we studied a subject-specific approach, in which the generalisation goal was to classify the task's condition from fNIRS data on an unseen part of the recording for each subject individually. This consisted in training, optimising, and evaluating each of the classifiers on each subject of each dataset individually. Therefore, the same procedure as the subject-independent approach was used but with each participant of each dataset individually. The only difference was that the outer and inner cross-validations consisted of stratified 5-fold and 3-fold cross-validations respectively instead of group k-fold, such that the class distribution remained balanced in the training, validation, and test sets.

4.3 Results

4.3.1 Subject-independent approach

The models are first compared to each other with the maximum number of training examples available and the maximum time window length of 10 seconds, which took 29 hours and 7 minutes to run with the configuration described previously.

The results for each dataset can be found in Figure 4.1 and Table 4.1.

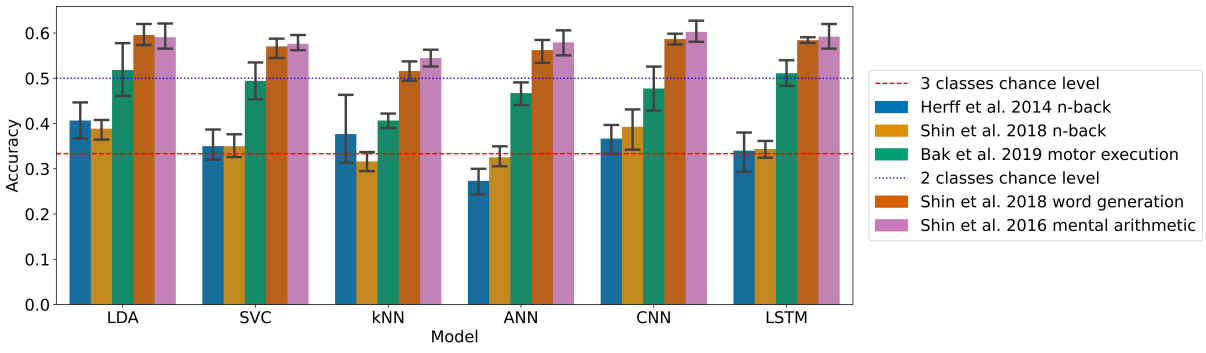

Figure 4.1: Accuracies of the models on each dataset with the subject-independent approach, models are all using HbO and HbR simultaneously (95% confidence intervals related to the variability on the 5-fold outer cross-validation are displayed).

Table 4.1: Accuracies of the models on each dataset with the subject-independent approach, models are all using HbO and HbR simultaneously. Fields marked with an asterisk indicate an accuracy significantly greater than chance level at a 5% threshold, the standard deviation on the 5-fold outer cross-validation is in parenthesis.

Dataset	**Chance level**	**LDA**	**SVC**	**kNN**	**ANN**	**CNN**	**LSTM**
Herff 2014	0.333	0.407 *	0.350	0.377	0.273	0.367	0.340
n-back		(0.049)	(0.038)	(0.083)	(0.033)	(0.038)	(0.049)
Shin 2018	0.333	0.389 *	0.350	0.316	0.325	0.393 *	0.344
n-back		(0.026)	(0.029)	(0.026)	(0.026)	(0.049)	(0.021)
Shin 2018	0.500	0.596 *	0.570 *	0.516	0.562 *	0.587 *	0.584 *
word generation		(0.026)	(0.026)	(0.027)	(0.028)	(0.013)	(0.008)
Shin 2016	0.500	0.591 *	0.576 *	0.545 *	0.579 *	0.602 *	0.592 *
mental arithmetic		(0.035)	(0.021)	(0.022)	(0.033)	(0.026)	(0.030)
Bak 2019	0.333	0.518 *	0.494 *	0.407 *	0.467 *	0.477 *	0.511 *
motor execution		(0.068)	(0.048)	(0.019)	(0.030)	(0.059)	(0.033)

On the n-back tasks from Herff et al. 2014 and Shin et al. 2018 the accuracy was found significantly higher than chance level (33.3%) with the LDA with p-values of 0.020 and 0.006 respectively, reaching with 3 classes 40.7% and 38.9% respectively. For the Shin et al. 2018 dataset of n-back tasks the CNN accuracy of 39.3% was also found significantly higher than chance level with a p-value of 0.037.

For the Shin et al. 2018 dataset of word generation tasks, significant differences compared to chance level (50%) were found for the LDA, SVC, ANN, CNN and LSTM with p-values of 0.001, 0.031, 0.005, <0.001 and <0.001 respectively. For those models, the accuracy ranges from 57.0 to 59.6% for traditional machine learning and from 56.2 to 58.7% for deep learning. A Wilcoxon test was used to test the significance on the SVC due to the non-normality of the distribution as measured with a Shapiro test.

For the Shin et al. 2016 dataset of mental arithmetic tasks, significant differences compared to chance level (50%) were found for all the models with p-values ranging from 0.001 to 0.004. A Wilcoxon test was used here to test the significance on the kNN due to the non-normality of the distribution as measured with a Shapiro test. The accuracies range from 54.5 to 59.1% for the machine learning models and from 57.9 to 60.2% for the deep learning models.

Finally, for the Bak et al. 2019 dataset of motor execution tasks, significant differences compared to chance level (33.3%) were also found for all the models implemented with p-values up to 0.004. The accuracies range from 40.7 to 51.8% for the machine learning models and from 46.7 to 51.1% for the deep learning models.

A statistical influence of the model on the accuracy was found for each dataset except the Shin et al. 2016 dataset of mental arithmetic tasks (Kruskal-Wallis tests were used for the Shin et al. 2018 dataset of word generation and the Shin et al. 2016 dataset of mental arithmetic because of non-normality). More specifically with pairwise t-tests, on the Shin et al. 2018 dataset of n-back tasks, the accuracy of the LDA was found significantly greater than the accuracy of the kNN. On the Shin et al. 2018 dataset of word generation tasks, the LDA and CNN accuracies were found significantly greater than the kNN. Finally, on the Bak et al. 2019 dataset of motor

execution tasks, the LSTM accuracy was found significantly greater than the kNN and ANN accuracies.

A detail of the hyperparameters selected with grid search for each iteration of the outer cross-validation can be found in the supplementary materials, as well as the results of the statistical tests.

4.3.2 Influence of the number of training examples (subject-independent approach)

The correlation between the number of training examples and the classification accuracy is shown, for each dataset and each model, in Table 4.2. This took 93 hours and 15 minutes to run.

Only two significant correlations were found with a threshold of 5%. The kNN on the Herff et al. dataset of n-back tasks was negatively influenced by an increase in training examples with a p-value of 0.039 and a correlation coefficient of -0.378. The other one was with the CNN on the Bak et al. 2019 dataset of motor execution tasks where the accuracy was positively influenced by the number of training examples with a p-value of 0.033 and a correlation coefficient of 0.390.

Table 4.2: Correlation coefficients of the relationship between accuracy and number of training examples. Fields marked with an asterisk indicate a significant correlation at a 5% threshold.

Dataset	**LDA**	**SVC**	**kNN**	**ANN**	**CNN**	**LSTM**
Herff 2014 n-back	0.111	0.181	-0.378 *	-0.340	0.043	-0.342
Shin 2018 n-back	0.087	0.026	0.083	-0.333	0.317	0.160
Shin 2018 word generation	0.148	-0.206	-0.080	-0.035	0.243	0.007
Shin 2016 mental arithmetic	-0.102	-0.120	0.128	-0.268	-0.123	-0.008
Bak 2019 motor execution	0.120	0.059	0.265	0.143	0.390 *	0.187

4.3.3 Influence of the time window length (subject-independent approach)

The influence of the time window length used as input of the models on the classification accuracy of the LDA, SVC, kNN, and ANN can be seen in Figure 4.2. This took 38 hours and 41 minutes to run.

For the Herff et al. 2014 dataset of n-back tasks, a significant positive correlation was found for the LDA (p-value of 0.004) with a correlation coefficient of 0.417 and a negative correlation for the ANN (p-value <0.001) with a correlation coefficient of -0.513.

For the Shin et al. 2018 dataset of n-back tasks, a significant negative correlation was found for the kNN with a correlation coefficient of -0.319 (p-value of 0.033).

For the Shin et al. 2018 dataset of word generation tasks, significant positive correlations were found for the LDA, SVC, and ANN with p-values inferior or equal to 0.001. The correlation coefficients are 0.732, 0.585, and 0.462 respectively. Spearman tests were used for the LDA and the SVC because of the non-normality of distributions.

For the Shin et al. 2018 dataset of mental arithmetic tasks, significant positive correlations were found for the LDA, SVC, kNN and ANN with p-values inferior to 0.001. The correlation coefficients are 0.510, 0.665, 0.504, and 0.646 respectively.

For the Bak et al. 2019 dataset of motor execution tasks, significant positive correlations were found for the LDA, SVC, kNN and ANN with all p-values inferior to 0.001. The correlation

coefficients are 0.803, 0.788, 0.618, and 0.836 respectively. A Spearman test was used for the LDA because of the non-normality of the distribution.

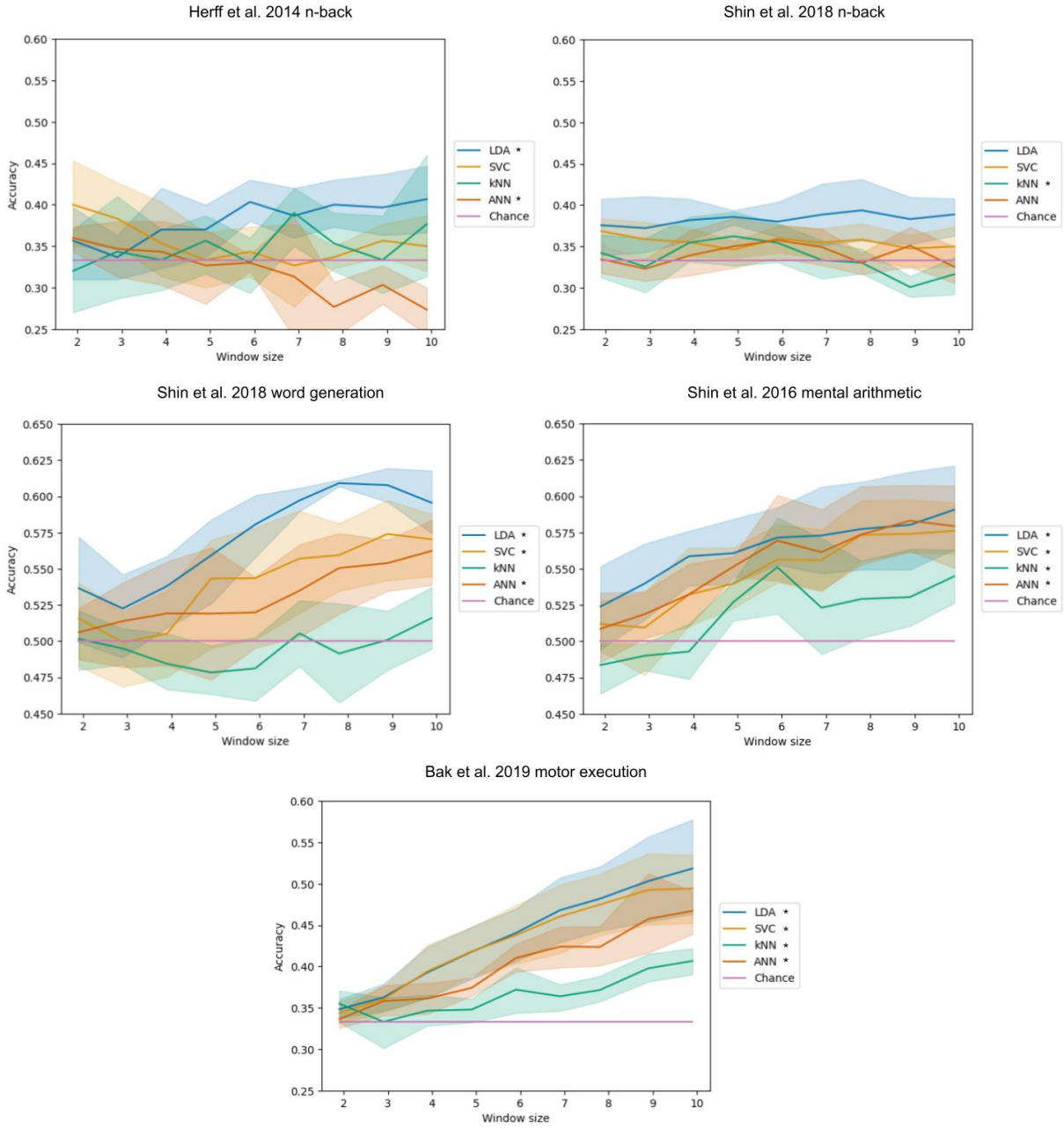

Figure 4.2: Accuracy with respect to the time window length. The bands represent the 95% confidence interval related to the variability on the 5-fold outer cross-validation. Legend entries marked with an asterisk indicate a significant correlation at a 5% threshold.

4.3.4 Subject-independent approach with sliding window

The results with the subject-independent approach using a 2 seconds sliding time window for the LDA, SVC, kNN and ANN can be found in Figure 4.3 and in Tables 4.3. This took 22 hours and 19 minutes to run.

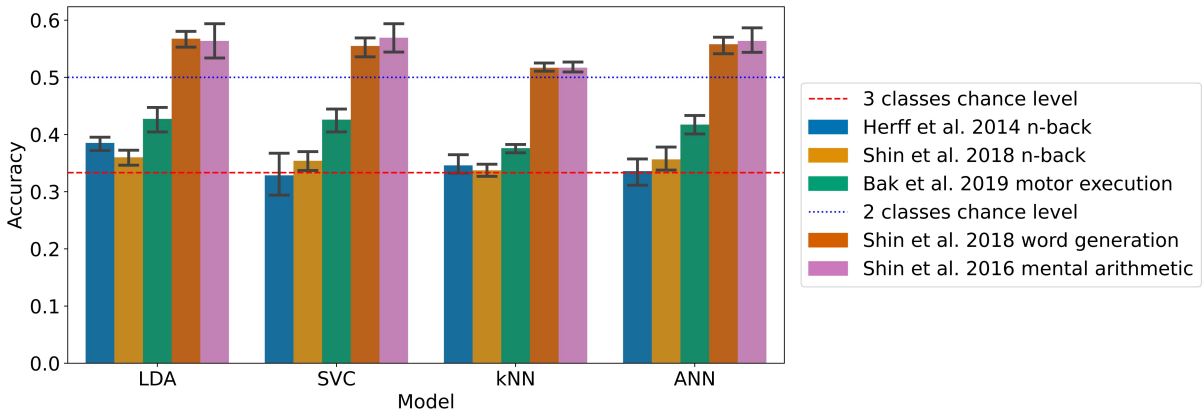

Figure 4.3: Accuracies of the models on each dataset with the subject-independent approach using a 2-second sliding time window, models are all using HbO and HbR simultaneously (95% confidence intervals related to the variability on the 5-fold outer cross-validation are displayed).

Table 4.3: Accuracies of the models on each dataset with the subject-independent approach using a 2-second sliding time window, models are all using HbO and HbR simultaneously. Fields marked with an asterisk indicate an accuracy significantly greater than chance level at a 5% threshold, the standard deviation on the 5-fold outer cross-validation is in parenthesis.

Dataset	**Chance level**	**LDA**	**SVC**	**kNN**	**ANN**
Herff 2014	0.333	0.385 *	0.329	0.346	0.336
n-back		(0.014)	(0.040)	(0.020)	(0.028)
Shin 2018	0.333	0.360 *	0.354	0.337	0.356
n-back		(0.014)	(0.021)	(0.013)	(0.023)
Shin 2018	0.500	0.568 *	0.555 *	0.517 *	0.558 *
word generation		(0.016)	(0.019)	(0.008)	(0.017)
Shin 2016	0.500	0.564 *	0.569 *	0.517 *	0.564 *
mental arithmetic		(0.036)	(0.028)	(0.011)	(0.024)
Bak 2019	0.333	0.427 *	0.426 *	0.376 *	0.417 *
motor execution		(0.024)	(0.023)	(0.009)	(0.019)

For the n-back task datasets from Herff et al. 2014 and Shin et al. 2018, the accuracy was found significantly greater chance level (33.3%) for the LDA with p-values of 0.001 and 0.010 respectively, corresponding to accuracies of 38.5 and 36.0% respectively with 3 classes.

For the Shin et al. 2018 dataset of word generation tasks, significant differences compared to chance level (50%) were found for the LDA, SVC, kNN and the ANN with p-values up to 0.008. The accuracies range from 51.7 to 56.8% with 2 classes.

For the Shin et al. 2016 dataset of mental arithmetic tasks, significant differences compared to chance level (50%) were found for the LDA, SVC, kNN and ANN with p-values ranging from 0.003 to 0.02 and accuracies ranging from 51.7 to 56.9% with 2 classes.

For the Bak et al. 2019 dataset of motor execution tasks, significant differences compared to chance level (33.3%) were found also for the LDA, SVC, kNN and ANN with p-values inferior to 0.001. The accuracies range from 37.6% with the kNN to 42.7% with the LDA.

A significant influence of the model on the classification accuracy was found for all the datasets except the Shin et al. 2018 dataset of n-back tasks. More specifically with pairwise t-tests, on the Herff et al. dataset of n-back tasks, the accuracy of the LDA was found significantly greater than the accuracy of the kNN. On the Shin et al. 2018 dataset of word generation tasks, the LDA accuracy was found significantly greater than all the other models with sliding window, and the kNN accuracy was found significantly lower than the SVC and ANN accuracies. Finally, on the Bak et al. 2019 dataset of motor execution tasks, the kNN accuracy was found significantly lower than all the other models with sliding window.

A detail of the hyperparameters selected with grid search for each iteration of the outer cross-validation can be found in the supplementary materials as well as the results of the statistical tests.

4.3.5 Subject-specific approach

The results with the subject-specific approach which took 24 hours and 58 minutes to run can be found in Table 4.4.

Table 4.4: Accuracies of the models on each dataset with the subject-specific approach, models are all using HbO and HbR simultaneously. The number of participants out of the total number of participants of each dataset having an accuracy significantly greater than chance level at a 5% threshold is presented in brackets for each model. The standard deviation on participants is in parenthesis.

Dataset	**Chance**	**LDA**	**SVC**	**kNN**	**ANN**	**CNN**	**LSTM**
Herff 2014	0.333	$0.353^{[1/10]}$	$0.350^{[1/10]}$	$0.350^{[4/10]}$	$0.373^{[2/10]}$	$0.353^{[0/10]}$	$0.360^{[2/10]}$
n-back		(0.099)	(0.065)	(0.128)	(0.076)	(0.034)	(0.099)
Shin 2018	0.333	$0.360^{[3/26]}$	$0.319^{[3/26]}$	$0.329^{[0/26]}$	$0.343^{[2/26]}$	$0.317^{[0/26]}$	$0.356^{[1/26]}$
n-back		(0.103)	(0.099)	(0.095)	(0.065)	(0.049)	(0.083)
Shin 2018	0.500	$0.588^{[7/26]}$	$0.562^{[6/26]}$	$0.546^{[4/26]}$	$0.562^{[8/26]}$	$0.554^{[8/26]}$	$0.549^{[6/26]}$
word generation		(0.090)	(0.099)	(0.068)	(0.088)	(0.089)	(0.076)
Shin 2016	0.500	$0.633^{[16/29]}$	$0.594^{[12/29]}$	$0.563^{[10/29]}$	$0.585^{[10/29]}$	$0.601^{[8/29]}$	$0.608^{[10/29]}$
mental arithmetic		(0.107)	(0.104)	(0.080)	(0.087)	(0.099)	(0.092)
Bak 2019	0.333	$0.513^{[18/30]}$	$0.444^{[14/30]}$	$0.380^{[7/30]}$	$0.387^{[6/30]}$	$0.402^{[8/30]}$	$0.446^{[15/30]}$
motor execution		(0.140)	(0.100)	(0.084)	(0.078)	(0.073)	(0.107)

The detailed statistical analysis for each subject can be found in the supplementary materials. The general trend however follows the one of the subject-independent approach, with more results significantly different from chance level with the dataset from Bak et al. 2019 of motor execution tasks and the dataset from Shin et al. 2016 of mental arithmetic. Moreover, the results are very subject dependant which is also shown by the high values of standard deviation as seen in Table 4.4.

4.4 Discussion

4.4.1 Subject-independent approach (*RQ2*)

Regarding the benchmarks of popular machine learning models on the 5 datasets, the first thing that the results show is that the performances are rather low overall (and typically lower than reported in some published works), which can be explained in multiple ways. First of all, the methodology prevents any kind of optimisation of the hyperparameters on the test set, which makes the results representative of what would happen with actual unseen data in the case of a real-life BCI application. Secondly, the models evaluated in this work present largely optimised baseline models that can be used in comparison for future machine learning developments and new datasets. More complex deep learning architectures, for example, could eventually help improve the performance. Also, the signal processing and the extraction of features have not been the focus of this work, and using approaches more personalised to each case would likely lead to better results for that case. Furthermore, most of the datasets found that meet our criteria are comparatively small, and research with more examples could be beneficial as we discuss in the following subsection, especially for the deep learning models. We hope that researchers will contribute both more advanced models and larger datasets to this benchmarking framework, as part of a shared community drive towards making clear advances.

Another finding is that the performances appear different with the type of task dataset. Indeed the performances on the Bak et al. 2019 dataset of motor execution tasks are higher than on the other datasets with 3 classes (Herff et al. 2014 and Shin et al. 2018 datasets of n-back tasks). An explanation could lie in the nature of the tasks: the brain activity elicited by motor execution is easier to highlight than the brain activity elicited by mental workload tasks which rely on higher-level brain processes [54]. We also see that the datasets with 2 classes of word generation (Shin et al. 2018) and mental arithmetic (Shin et al. 2016) also have classification accuracies generally greater than chance level, which may be explained by the fact that both are task detection datasets (baseline vs. task) as opposed to classifying the level of a task as done with the n-back datasets.

It also appears that the variability with different test sets can sometimes be quite high for some models and some datasets, which could mean that it is more difficult to perform classification on some unseen subjects compared to others, probably in cases where their data looks different than the one from participants in the training set.

Regardless, one of the outcomes is that in most cases the kNN seems to underperform compared to other models, while the other traditional machine learning models with feature extraction, especially the LDA, do not perform worse than deep learning methods using raw data (CNN and LSTM). Traditional machine learning models (LDA and SVC) then remain a relevant choice especially because of their low computational complexity. This goes in the same direction as other works such as [65] showing that deep learning methods achieve comparable accuracies to conventional methods.

4.4.2 Influence of training set size on subject-independent approach (*RQ3a*)

Only 2 significant correlations of the accuracy to the percentage of training examples have been found out of the 30 tested in total, which is quite low. Those are a negative correlation with a kNN model and a positive correlation with a CNN model. This positive correlation goes with the tendency of deep learning models to perform better with bigger datasets, however in our case, it has only been highlighted with one deep learning model on the motor execution dataset from Bak et al. 2019. Even though this dataset is amongst those with the most examples with 750 per class in total (leading in the training set to 600 examples per class with 100% and 300 per class with the minimum studied here of 50% of training data), this trend is not shown with the biggest dataset containing 870 examples per class (mental arithmetic from Shin et al. 2016). It is likely

that even with 100% of the training data, the training sets remain very small for all the datasets, especially for classification with deep learning. Indeed with 3 classes, the number of trainable parameters is 155, 491, and 17635 for the ANN, CNN and LSTM respectively (150, 480 and 17618 respectively with 2 classes) which is to relate to the total number of examples in the datasets between 300 and 2250. The influence of the dataset size would be interesting to study further with bigger fNIRS datasets to see if there is really a clear effect with deep learning models.

4.4.3 Influence of window length on subject-independent approach (*RQ3b*)

Based on the results, it appears that overall the length of the time window used as input for classification does have an influence on the performance of models using feature extraction, as the correlations show.

Each of our correlations is positive except for 2 negative correlations which were found on the n-back datasets. Those negative correlations are however hard to interpret with certainty since the accuracy on those n-back tasks remains very low overall and in most cases is not significantly higher than chance level as seen with the subject-independent approach with 10-second epochs. Also, precautions should be taken for short time windows on n-back tasks because they do not enable to express the whole difficulty of the task since a new stimulus is presented every 2 seconds from the beginning of the task and n stimuli are required to reach the actual task demand with n-back.

All other correlations were positive, for most of the models with which the classification accuracy was significantly greater than chance level, which makes us believe that a bigger time window actually benefits classification accuracy. This is especially striking for the Bak et al. 2019 dataset of motor execution tasks as seen in Figure 4.2. This benefit of longer time windows is likely explained by the duration of the hemodynamic response, being around 4 to 6 seconds [26], making shorter time windows too small to capture relevant changes.

4.4.4 Subject-independent approach with sliding window (*RQ3c*)

Using a sliding approach has two main advantages. First, it enables us to multiply the number of examples that can be used as input for the classifiers. Secondly, it enables us to make a prediction on the class every 2 seconds which can be useful in the context of a BCI.

With the models using feature extraction, the accuracies using a 2-second sliding window are found significantly greater than chance level in the same cases as the subject-independent approach with non-sliding 10-second epochs. Here again, the kNN seems to underperform in most cases compared to other models. For all the models tested here with sliding window, results appear overall slightly lower than those with the 10-second epochs, even though the performances are higher than those obtained with a non-sliding window of 2 seconds which are around chance level. It may be possible that the decrease in performance observed with smaller time windows as described previously is compensated by the increase in training examples. Also, even though previous work has shown potential at classifying fNIRS data using short time windows from the task trigger onset, hence focusing on the initial dip part of the hemodynamic response [87, 179], it may be possible that later segments are more useful to discriminate between conditions. Further studies would need to be conducted to compare different centring of shorter time windows as an extension of *RQ3b* and *RQ3c*. Regardless, such sliding window approaches remain relevant for BCI applications when it is desired to have predictions made regularly in real time.

4.4.5 Subject-specific approach (*RQ3d*)

The results highlight a very high variability across subjects, and the average results are not much different from the subject-independent approach. This subject-specific approach, though, seems to produce relatively high results on the Shin et al. 2016 dataset of mental arithmetic

tasks. It also appears that the LDA model with feature extraction performs quite well on most of the datasets.

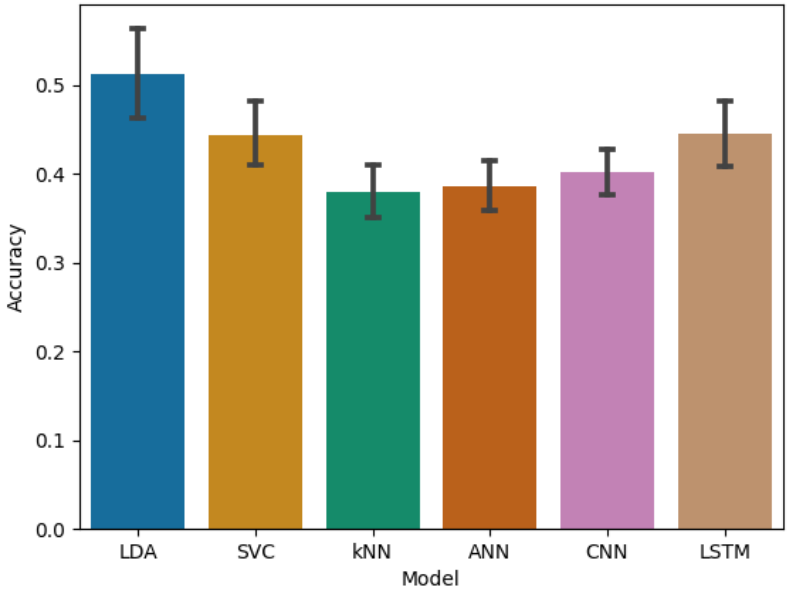

Figure 4.4: Accuracy for the subject-specific approach on the dataset from Bak et al. 2019 of motor execution tasks (95% confidence intervals related to the variability on the subjects are displayed)

It is interesting to note that for the dataset enabling the best model performances, while we had deep learning models performing the best with the subject-independent approach, in the subject-specific approach traditional machine learning models appear to perform the best (Figure 4.4). This could be related to the number of examples in each case and could be supporting evidence that we notice deep learning performing best with more data, and traditional machine learning performing best with small amounts of data.

These results with our methodology however remain low compared to what can be found in existing literature. Indeed, most of the papers that have proposed a classification using machine learning on the same datasets we used are using a subject-specific approach. For example, [66] reached slightly higher accuracies on their n-back dataset with a LDA using the slope on each channel and on a slightly longer time window (15 seconds) than our experiments, with 44.0% on average with 10-fold cross-validation (meaning more training data than 5-fold) compared to 35.3% with our baselines. [154] with shrinkage LDA using average and slope for each channel on 3 seconds moving windows reached 80.7% with HbR and 83.6% with HbO with 5-fold cross-validation on their mental arithmetic dataset, compared to 63.3% with our baselines. [7] with linear SVC using the average for each channel on 5 seconds moving time windows reached an average accuracy of 70.4% on their motor execution dataset, compared to 44.4% in our results. Finally, compared to our results around chance level with a CNN using temporal convolutions, [146] reached an accuracy of 82% on average on the Shin et al. 2018 dataset of n-back tasks with a CNN using spatial convolutions, however, they did not describe how the dataset was split into training, validation and test sets. All those existing results show the difficulty of comparison when lacking standardised methodology, also when some methods cannot be applied to other datasets due to constraints from the experimental design (eg. length of time windows, number of channels). We hope that *BenchNIRS* will allow future work with notably high accuracy to now more easily

demonstrate their advances for improved performance in a comparable way, and more easily check against common mistakes.

Another point of discussion is that the results of the subject-specific approach do not, unfortunately, give the opportunity to draw higher-level conclusions, as every subject only took part in one session in all the datasets, making it impossible to determine whether the results are related to session-specific factors or subject specific factors.

Finally, it should be noted that such subject-specific models are hardly usable in the context of a real-life BCI, especially if they require model training on each session. Indeed, in the case where the results are obtained with a 5-fold outer cross-validation, it means that the model requires 80% of the data for training, meaning that the majority of the time would be dedicated to calibrating the BCI with the subject rather than using it. This is also why our work mainly focuses on subject-independent approaches which can be applied more easily in real-life BCI settings. Also, this relates to why the data is not normalised with a min-max feature scaling or standard score: computing those on the whole dataset would bias the results on the test set (because test data would have been seen already to normalise) and is not possible in real time, and computing those with the mean or min-max computed on the training set only could shift the distribution of the test set if it is very different from the training set (if the mean or min-max are quite different on the test set than the training set).

4.4.6 Limitations and next steps

Our work provides novel insights into factors influencing the performance of machine learning classifiers using fNIRS data, in the hope of helping readers look for the model that would best suit their needs. This is however an entry point into the benchmarking of machine learning for fNIRS. Therefore, some limitations remain for future work to address.

First of all, there are limitations due to the datasets used in this framework. Most of the datasets contain a limited amount of data which is critical for the performance of some machine learning models, especially those having a lot of parameters. Datasets with more subjects could be added to the framework in the future, such as [73], however, the lower sampling frequency of this dataset in particular would have required downsampling all the other datasets for comparing results rigorously. Another point is that none of the datasets have used an fNIRS device with short-separation channels, which limits the extent of noise removal that can be performed. Indeed, having such short-separation channels for each dataset would have helped remove artefacts due to superficial hemodynamics reflecting systemic physiological changes [20, 148]. Also, the datasets were based on recordings from participants who took part in only one session, which limits the conclusions that can be drawn when it comes to studying participant or session specificity. Those constraints come with the limited availability of open access fNIRS datasets, and future work would consist of extending the framework to other newly published open-access datasets addressing those issues. Indeed, we welcome dataset contributions (see Chapter 3).

Secondly, the performance of our models may be limited by the compromises that had to be made for the sake of comparison between datasets with different experimental designs and different equipment. For example, we had to accommodate for the sampling frequency (which is why downsampling has been performed), the number of channels (which is why region of interest averaging has been performed), and epoch duration (which is why epoch cropping was performed).

Also, the work is also potentially limited by the signal processing and feature extraction used. The signal processing selected in our framework follows a recommended approach with TDDR and bandpass filtering to remove signal noise. This choice was made because the comparison of signal processing is not the focus here and has been studied in other published works such as [134] and [19]. However, the benchmarking could be extended by involving different and more advanced signal processing techniques. Indeed, approaches more tailored to each task would be more efficient at removing signal noise for that dataset, but a classic approach was used here for comparison. Similar remarks can be made regarding the feature extraction. Further, as the

datasets involved devices with different numbers of channels, we needed to average the channels in regions of interest for the sake of comparability of models, but more work could be done regarding the spatiality of the brain activity.

Finally, each of the machine learning models used in our study could have been developed and tweaked in different ways. We decided to implement common baseline models to act as a starting point for benchmarking, however, more complex models could be implemented. For example here, shrinkage LDA could be compared to standard LDA. Also, the architectures of the deep learning models have been chosen keeping into consideration the input data dimensions and the number of training examples, but they could be more extensively optimised, and finding optimal architectures could be a matter of future work. This could include, for example, architectures valuing more the spatiality of signals. Similarly, different kernels could be tested for the SVC for example. Our work, however, means that is now possible to implement such more advanced models in future work, using our framework, and for them to be validated robustly with the recommended checklist of methodological steps.

4.5 Summary

The *BenchNIRS* framework is used to perform the analysis on 5 open-access datasets of 6 baseline machine learning models: LDA, SVC, kNN, ANN, CNN and LSTM. We also used *BenchNIRS* to produce results with different approaches: subject-independent, subject-independent with a sliding window, and subject-specific. Further, we studied the influence of the training set size as well as the time window length (from 2 to 10 seconds) on the model performances.

To our knowledge, this multi-model multi-dataset benchmarking with the study of multiple influencing factors is the first of its kind for fNIRS, and the scale of it is thus far unrivalled, providing a large variety of objective insights for machine learning with fNIRS.

Where most published research has studied specific models applied to specific datasets, we show with our initial benchmarks that no baseline model (traditional machine learning or deep learning) statistically stands out consistently compared to others when applied across datasets except the LDA, hence remaining a strong choice despite its simplicity. Furthermore, we found no consistent influence of the training size on the classification accuracy. Finally, our results show that when models using common feature extraction techniques (mean, standard deviation, slope) perform greater than chance level, they benefit from longer time windows.

We hereby validate the usability of *BenchNIRS* (*RQ1*) for the purpose of evaluating and comparing different machine learning approaches with fNIRS data. We invite the fNIRS community to use the framework when performing classification with their own machine learning models for a convenient comparison to our initial baseline results.

Chapter 5

Tailored supervised machine learning

- *RQ4*: How do machine learning approaches tailored to a specific mental workload task compare to baseline benchmarks?
  - *RQ4a*: What are the benchmarks of baseline machine learning models tailored for n-back tasks?
  - *RQ4b*: How does increasing further the time window duration of the examples influence the models' performances?
  - *RQ4c*: How can tailoring a supervised deep learning classification model to a specific n-back dataset improve its performance?

5.1 Task-tailored baselines

5.1.1 Motivations

After having produced results for some baseline models, designed to work on all datasets in Chapter 4, we described how those baseline benchmarks were limited by some choices that had to be made to limit the number of variables in the comparison of models on all the datasets.

More specifically in this section we will be interested in overcoming a first limitation which involved the epoch duration. For the purpose of comparison of the models on all the datasets, in Chapter 4 we had to crop the epoch duration to only 10 seconds as it was the shortest block duration of all the datsets, however some datasets had quite longer block duration, such as the n-back tasks [66, 155]. This was a limitation as we saw also that the time duration of examples did influence the performance of models and that it was positively correlated with accuracy, except the n-back datasets, which we suspected being related to the low performance of the models on those tasks. Here we also want to question whether the fact that the epochs of those tasks were the most reduced could have also affected this correlation and overall the results of machine learning models. This is the first reason why here we will use this specific n-back task to re-run the baseline benchmarks with longer epoch duration.

Another reason for using this task is that it is, as we described in Chapter 2, one of the most common tasks used when studying mental workload in experimental settings, especially for neuropsychology studies [125]. This makes it a good toy task for applications for mental workload assessment with fNIRS BCIs. n-back task level classification from fNIRS signals at a reliable level remains a difficult achievement as we saw before (Chapter 4), however this is a very interesting challenge for the future of fNIRS in passive BCIs for mental workload assessment [10].

Here we will put the focus on performing task classification on unseen subject, as discussed in Chapter 4, for investigating the simplest case close to realistic BCI applications: subject-specific approaches as studied before require too much calibration (training) data proportionally to the session duration to be usable in practice. This use case also, as opposed to sliding windows, relies

on an established neuroscientific basis, namely capturing the hemodynamic response function (HRF) on block design tasks [103, 176].

For this section, we will therefore ask the following research question:

- *RQ4*: How do machine learning approaches tailored to a specific mental workload task compare to baseline benchmarks?

More specifically we decompose here this into 2 sub-questions:

- *RQ4a*: What are the benchmarks of baseline machine learning models tailored for n-back tasks?
- *RQ4b*: How does increasing further the time window duration of the examples influence the models' performances?

To answer those questions, we will first use the *BenchNIRS* framework to present results of baseline models on the n-back task datasets (from Herff et al. published in 2014 [66] and Shin et al. published in 2018 [155]). Secondly, we will use it to also present extended results of the influence of time window duration on the longer n-back task blocs.

5.1.2 Methods

We use here a similar methodology to evaluate the same models as the initial benchmarks (LDA SVC, kNN, ANN, CNN and LSTM) but adapted for n-back tasks specifically, meaning that we use longer epoch duration.

Datasets

The 2 n-back open access datasets supported by *BenchNIRS* were used for this investigation: the one from Herff et al. [66] and the one from Shin et al. [155] described previously in Chapter 3. The n-back dataset from Herff et al. 2014 was collected with a device of 8 channels for each wavelength, and comported of 3 classes with 100 examples each in total: 1-back, 2-back and 3-back; while the n-back dataset from Shin et al. 2018 was collected with a device of 36 channels for each wavelength and was also composed of 3 classes with 234 examples each in total: 0-back, 2-back and 3-back.

fNIRS processing and feature extraction

The HbO and HbR data for the 2 datasets were processed similarly to Chapter 4, using *BenchNIRS*. This included TDDR motion artefact correction, bandpass filtering with cutoff frequencies of 0.01 and 0.5 Hz, and region of interest averaging by hemisphere to end up with processed HbO and HbR for left and right PFC (process illustrated in Figure 3.1). *BenchNIRS* also enabled to extract epochs using the onset triggers up with an epoch duration of 40 seconds (shortest task duration, from [66, 155] having a task duration of 44 sec). The 2 seconds prior to each trigger onset were used for baseline correction, and the data was downsampled to 10 Hz to then be converted into µM. This resulted in each example having a shape of 4x400.

Feature were extracted following the methodology described in the initial benchmarks for the traditional machine learning models (LDA SVC and kNN) and also the ANN. Those extracted features were the mean, standard deviation and slope of linear regression across time for each region of interest.

Machine learning models and methodology

The same baseline models as before were used for evaluation: LDA SVC, kNN, ANN, CNN and LSTM. The only differences being for 2 of the models:

- for the LSTM, instead of having the input arranged as a sequence of 5 elements of 2 sec, they were arranged as a sequence of 22 elements of 2 sec;

- the CNN, using signal processed data without feature extraction, was composed of 2 one-dimensional convolutional layers across the time axis: the first one with 4 input channels and 4 output channels, a kernel size of 10 (one dimension kernel) and a stride of 2; the second one with 4 input channels and 4 output channels, a kernel size of 5 (one dimension kernel) and a stride of 2. The first convolutional layer was followed by a one-dimensional max pooling across the time axis with a kernel size of 2, and the second convolutional layer was followed by the same max pooling but with a kernel size of 4 instead. Those convolutions and max poolings were followed by 2 fully connected layers of 44 and 24 neurons respectively, followed to finish with by an output layer of 3 neurons, matching the number of classes for the dataset. All the activation functions of the hidden layers were rectified linear units (ReLU), the loss function and optimiser were cross-entropy loss and Adam respectively.

The models were evaluated on each dataset separately. *BenchNIRS* was used to perform a nested group k-fold cross-validation, with 5 outer folds (extraction of test sets) and 3 inner folds (extraction of validation sets), in order to split the dataset without mixing subjects. The validation sets were used for hyperparameter optimisation while the test sets were used for evaluation of the models. The goal was to evaluate the generalisation capabilities of the models at task classification from fNIRS data on unseen subjects, following the same approach as the subject-independent approach described in Chapter 4.

The optimisation of hyperparameters was done within the following ranges [12] with grid search using *BenchNIRS* [11]:

- the SVC's regularisation parameter values tested were 0.001, 0.01, 0.1 and 1;

- the values of k (number of nearest neighbours) tested were the integers from 1 to 9;

- the learning rate's values tested for the deep learning models were 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1};

- the mini-batch sizes tested for the deep learning models were 4, 8, 16, 32 and 64.

After the best set of hyperparameters was found the model was retrained on the whole training and validation set, and early stopping was performed for the deep learning models leaving out randomly 20% for a new validation set used to stop training before the maximum number of epochs of 100 if the loss on it was increasing for 5 consecutive epochs.

Statistics were computed in order to compare the accuracy of each model (metric selected because classes were balanced for each dataset) on the test sets to chance level. This involved a one-tailed t-test to determine whether the accuracy was greater than chance level if the distribution of the accuracies of the model on the test sets followed a normal distribution as tested with a Shapiro test; otherwise a one-tailed Wilcoxon test was used.

A confusion matrix was produced for each model of each n-back dataset, comparing the predictions made across all the outer folds against the true class labels (those can be found in the appendices).

Moreover, new models with 40-second epochs implemented here were also compared to the old models with 10-second epochs presented in Chapter 4 for the two n-back datasets. This was done using one-tailed t-tests if normality and homogeneity of variances were verified using Shapiro and Bartlett tests, otherwise a one-tailed Mann-Whitney U was used.

In a second part, the influence of epoch duration was studied like before and we compared durations of 5, 10, 15, 20, 25, 30, 35 and 40 seconds from the trigger onset. For each duration tested, the methodology was the same as described previously. Correlation of the model accuracy

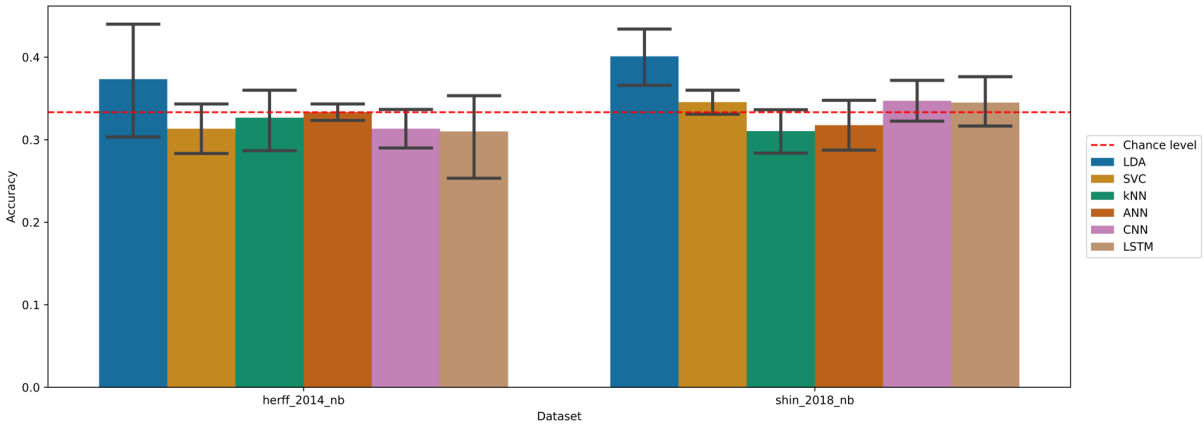

Figure 5.1: Accuracies of the models on the n-back datasets with the subject-independent approach (95% confidence intervals related to the variability on the 5-fold outer cross-validation are displayed).

to the duration was studied with Pearson's correlation test for each model of each dataset if the assumption of normality was verified as per a Shapiro test, otherwise a Spearman's correlation test was used. This study on the influence of epoch duration was performed on the LDA SVC, kNN, ANN, and LSTM. The CNN was not included in this analysis since its architecture depends on the input data shape which would be different for each epoch duration.

The threshold of significance was set at 5% for every statistical analysis. The implementation of this machine learning experiment can be found in the supplementary materials.

5.1.3 Results

Subject-independent approach

First, we can see the accuracy of each model on the n-back datasets for an extended epoch duration of 40 seconds (instead of 10 seconds in the initial baseline benchmarks) in Figure 5.1 and Table 5.1. We see once again that the performance is around the chance level of 33.3%.

Table 5.1: Accuracies of the models on the n-back datasets with the subject-independent approach. Fields marked with an asterisk indicate an accuracy significantly greater than chance level at a 5% threshold, the standard deviation on the 5-fold outer cross-validation is in parenthesis. A dagger mark signifies a significant increase in accuracy compared to the initial baseline model while a double dagger mark signifies a significant decrease.

Dataset	**Chance level**	**LDA**	**SVC**	**kNN**	**ANN**	**CNN**	**LSTM**
Herff 2014	0.333	0.373	0.313	0.327	0.333†	0.313‡	0.310
n-back		(0.075)	(0.034)	(0.040)	(0.011)	(0.029)	(0.056)
Shin 2018	0.333	0.401*	0.345	0.310	0.318	0.347	0.345
n-back		(0.040)	(0.017)	(0.031)	(0.036)	(0.029)	(0.034)

More specifically, on the n-back dataset from Herff et al. 2014, none of the models performed significantly better than chance level. For the n-back dataset from Shin et al. 2018, we see that only the LDA, with an accuracy of 40.1% performed significantly better than chance, with a p-value of 0.014.

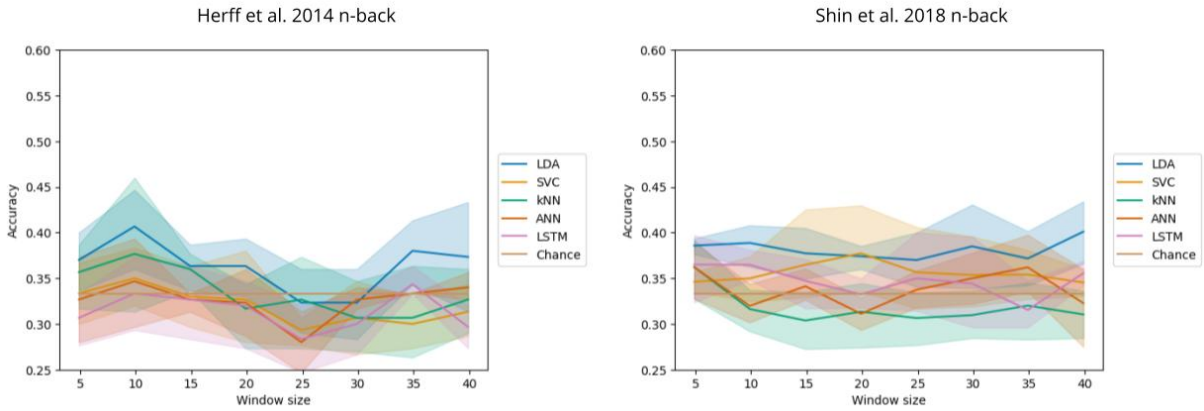

Figure 5.2: Accuracy with respect to the time window length. The bands represent the 95% confidence interval related to the variability on the 5-fold outer cross-validation. Legend entries marked with an asterisk indicate a significant correlation at a 5% threshold.

When comparing those new models using 40-second epochs to the original baseline models using 10-second epochs, we do not have any significant difference for the Shin et al. 2018 n-back dataset. Indeed, the original LDA also had an accuracy significantly greater than chance level then. But, while the original CNN performed significantly better than chance level initially, in this new experiment with 40-second epochs it does not anymore, however this decrease of accuracy between the new and original models is not significant. We only observe significant differences between the original and new models on the Herff et al. 2014 n-back dataset, where the new ANN with an accuracy of 33.3% performs better than the original one (p-value of 0.004) which had an accuracy of 27.3%, however this is hard to interpret reliably since this accuracy is actually lower than chance level. Also, the new CNN with an accuracy of 31.3% performed worse than the initial one (p-value of 0.028) which had an accuracy of 36.7%. Again, this is hard to interpret since none of them had an accuracy significantly greater than chance level.

A detail of the hyperparameter selection with grid search and the results of the statistical tests run for comparisons can be found in the supplementary materials.

Influence of window size

Then, the influence of the window size or epoch duration used as input of the machine learning models was studied from 5 to 40 seconds. The results are presented in Figure 5.2. Once again, we can see that the results are close to chance level, regardless of the window size. Statistical tests found correlations between accuracy and window size only on the Herff et al. 2014 n-back dataset. On that dataset, we found a negative correlation with a p-value of 0.04953 for the SVC (correlation coefficient of -0.313) and for the kNN with a p-value of 0.0245 (correlation coefficient of -0.355). Spearman tests were used for those two correlations because of the non-normality of the distribution. Those negative correlation coefficients mean that the accuracy was actually decreased when the time window size increased.

This is in line with the findings with the subject-independent approach that indicate that increasing from 10 to 40 seconds the window did not necessarily enable improvements. Here we see that increasing the window size actually appears to be detrimental to n-back level classification performance.

5.2 Dataset-tailored model

5.2.1 Motivations

A second limitation that we will be interested in overcoming here was the fact that the datasets were recorded with different fNIRS devices and that region of interest averaging had to be performed in order to end up with the same number of channels for each dataset. Even though this process can be useful in cases where optodes are not precisely positioned and can also help mitigate the effect of a noisy channel, this process is not ideal as it reduces the spatial resolution of the data.

So after having investigated in Section 5.1 the tailoring of baseline models to a specific task, n-back, we here aim to go slightly further and focus on one model without region of interest averaging of channels on one dataset. We here design the architecture of a model specifically for that dataset's task and acquisition device.

Following up on Section 5.1, we will here still be interested in n-back tasks for the motivations listed previously, but will more specifically use only one dataset: the dataset of n-back tasks from Shin et al. [155], and the goal will be to design a CNN with an architecture tailored for this dataset.

This combination of dataset with model is selected for multiple reasons. First, we showed in previous work [10] that deep learning had the potential to classify mental workload tasks from portable fNIRS devices, and those types of models also have the highest degree of customisation, compared to traditional machine learning, as their architecture can be adapted. Secondly, the CNN model was the only deep learning model that had an accuracy significantly higher than chance level in the initial benchmarks, with 39.3% accuracy for 3 classes. Thirdly, this type of model benefits from using data with less feature extraction (region of interest averaging can be considered as feature extraction), as convolutions enable to extract features within the model. Furthermore, the dataset chosen is the one with the most examples per class, which usually benefits deep learning models.

Similarly to before, we will here still be interested in task classification from fNIRS data on unseen subjects.

We will therefore here continue the investigation in the the overall research question for this chapter:

- *RQ4*: How do machine learning approaches tailored to a specific mental workload task compare to baseline benchmarks?

More specifically, in this section we will be interested in:

- *RQ4c*: How can tailoring a supervised deep learning classification model to a specific n-back dataset improve its performance?

This investigation will act as a further validation of *BenchNIRS* (*RQ1*) and present a concrete use case for the framework. It will tell us how the framework can be used to help the investigation of new deep learning architectures on a specific task.

5.2.2 Methods

Dataset

For the purpose of this dataset-tailored model implementation, the dataset of n-back tasks from Shin et al. was used [155]. This dataset has been described previously and more information can be found about it in Chapter 3. This dataset used a NIRScout device from NIRx Medical Technologies, with wavelengths of 760 and 850 nm and a sampling rate of 10 Hz. From its 16 sources and 16 detectors resulting in 36 channels, 16 channels around the PFC were used for each HbO and HbR (Figure 5.3), with a source-detector distance of 30 mm. Only those

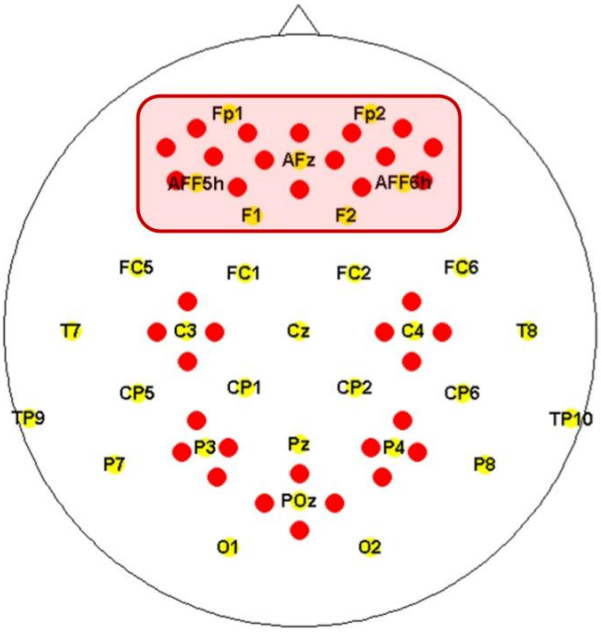

Figure 5.3: 16 fNIRS channels (red dots) selected for the dataset-tailored investigation on the n-back task from Shin et al. 2018, located in the red frame area around Fp1, Fp2, AFz, AFF5h, aAFF6h, F1 and F2 .

channels in PFC were selected as this is the region of interest described to reflect the type of brain activity elicited by working memory and mental workload tasks such as n-back [54, 119]. As described before, the tasks consisted in 0-back, 2-back and 3-back, with 234 examples per class (26 participants in total).

fNIRS processing and feature extraction

The signals are processed similarly to Section 5.1 using *BenchNIRS*: TDDR, bandpass filtering (0.01-0.5 Hz) and conversion into µM. Epoch duration was 40 seconds from the trigger onset, like before, which is the total duration of the task on the n-back dataset from Shin et al. 2018, and 2 seconds before the trigger onset were used for baseline correction. However, the 16 channels for each chromophore are not averaged by regions of interest, in order to adapt the model specifically for this optode configuration. No further features were extracted from the signals, so the shape of each input example was 32x400, representing 16 channels of HbO and 16 channels of HbR by 400 timepoints (40 seconds of 10 Hz signals).

Deep learning model and methodology

The model used here is a CNN (Figure 5.4) that was composed of 2 one-dimensional convolutional layers across the time axis: the first one with 32 input channels and 12 output channels, a kernel size of 16 (one dimension kernel) and a stride of 4; the second one with 12 input channels and 4 output channels, a kernel size of 8 (one dimension kernel) and a stride of 3. Each convolutional layer is followed by a one-dimensional max pooling across the time axis with a kernel size of 2. Those convolutions and max poolings were followed by 3 fully connected layers of 28, 16 and 8 neurons respectively, the second and third followed by a 20% dropout. Finally, there is an output layer of 3 neurons, matching the number of classes for the dataset. All the hidden layers activation functions were rectified linear units (ReLU), and we used the Adam optimiser with a cross-entropy loss function. The architecture proposed here has been implemented with PyTorch [126].

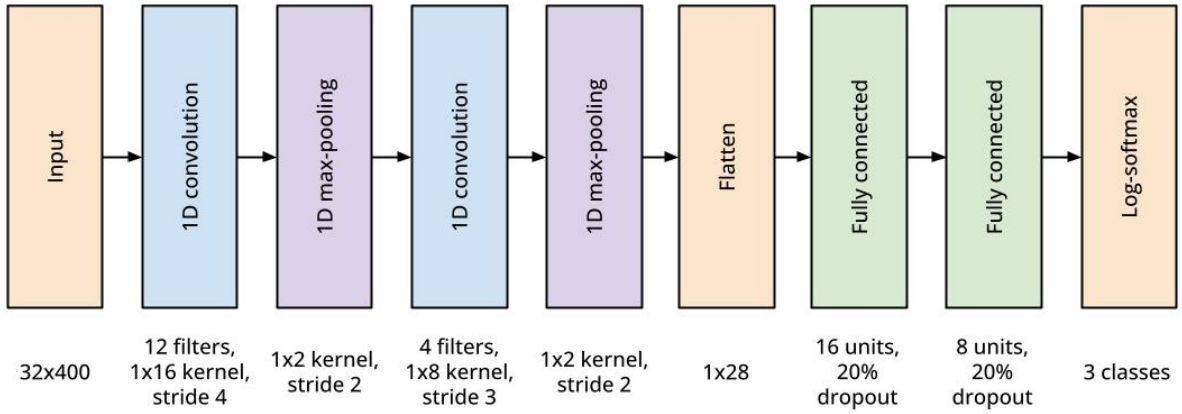

Figure 5.4: CNN architecture.

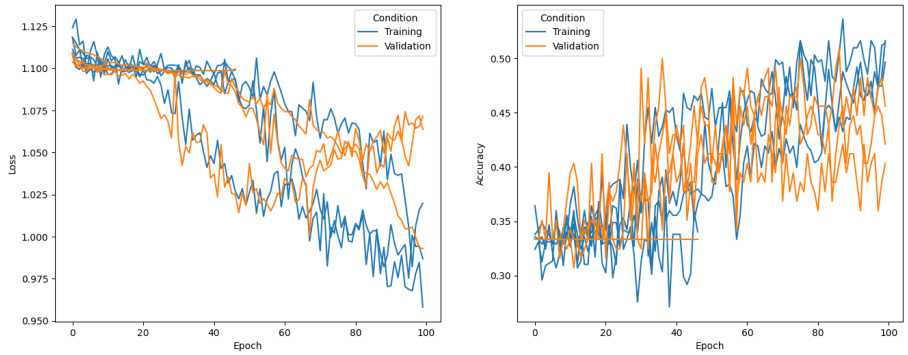

Figure 5.5: Training and validation graphs of the dataset-tailored CNN (cross-entropy loss on the left and accuracy on the right).

The same nested group k-fold cross-validation from *BenchNIRS* was used for optimisation of hyperparameters and model evaluation of the generalisation capabilities on data from unseen subjects.

The input data was normalised per channel with min-max scaling using minimums and maximums on the training and validation sets.

Hyperparameter optimisation was performed on the validation sets with grid search in the following ranges [12]:

- the learning rate's values tested for the deep learning models were 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1};
- the mini-batch sizes tested for the deep learning models were 4, 8, 16, 32 and 64.

Validation sets were also used to select the model architecture thanks to the training graphs, by attempting to have all the validation losses the lowest possible.

After fine-tuning the model to select the best set of hyperparameters, it was retrained with early stopping similar to Section 5.1: 20% of validation set randomly left out, 100 epochs maximum, and patience on the validation loss of 5 epochs. Training graphs can be seen in Figure 5.5.

We ran a one-tailed t-test or one-tailed Wilcoxon test (depending on the normality of distribution of the accuracies on the test sets) to determine whether the model had an accuracy significantly greater than chance level (accuracy was the selected metric since classes were perfectly balanced).

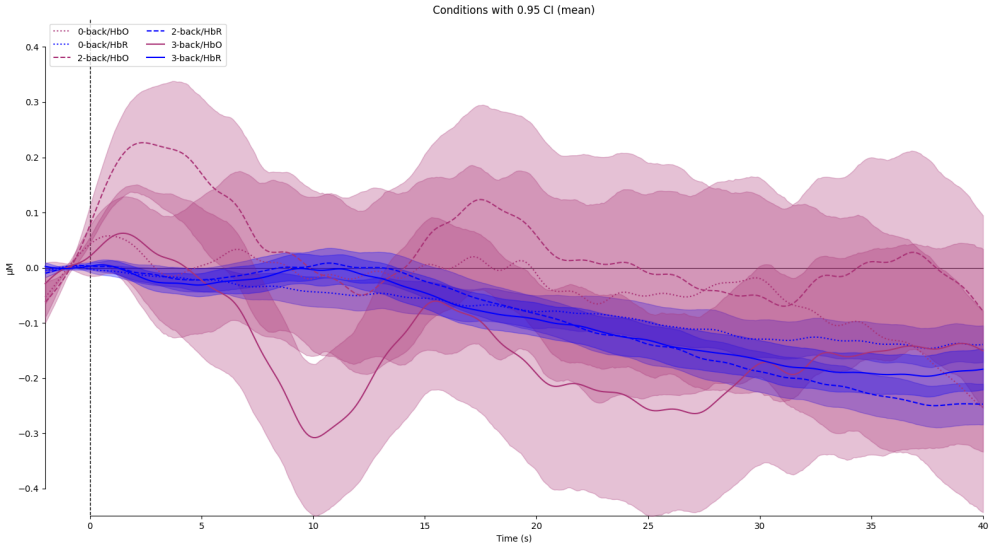

Figure 5.6: Grand average of the epochs of each n-back condition for HbO and HbR. The bands represent the 95% confidence interval related to the variability on the epochs of each condition.

Finally, we compared the performance of the dataset-tailored CNN also to the task-tailored CNN and LDA (only model performing significantly better than chance level) implemented in Section 5.1, and to the initial subject-independent baseline CNN for the same dataset. This was done using one-tailed t-tests if normality and homogeneity of variances were verified using Shapiro and Bartlett tests, otherwise a one-tailed Mann-Whitney U was used.

The implementation of this machine learning experiment can be found in the supplementary materials.

5.2.3 Results

To have an idea of the signal shape for each condition, we visualised the grand average of the epochs with all channels averaged for each condition in Figure 5.6. We see here that the 2-back condition specifically seems to present a stronger hemodynamic response pattern in HbO compared to the 2 other conditions (0-back and 3-back). We also see from the confidence intervals that the variability between the different epochs of each condition is quite important.

The CNN model working with inputs of 32 channels and 400 timepoints ended up with an accuracy of 39.9% on average on the 5 outer fold left-out test sets of the cross-validation with a standard deviation of 5.5%. This was significantly greater than chance level (33.3% with 3 classes) with a p-value of 0.037. The confusion matrix is presented in Figure 5.7, where we can see that the most accurately predicted class is 2-back. The hyperparameter optimisation led to a learning rate of 0.01 being selected for each iteration of the outer 5-fold cross-validation and a min-batch size of 8 selected once, 32 selected twice, and 64 selected twice.

When then went on to compare this new CNN tailored to the Shin et al. 2018 n-back dataset to previous models using region of interest averaging:

- the task-tailored LDA from Section 5.1 which reached an accuracy of 40.1%;
- the task-tailored CNN from Section 5.1 reaching an accuracy of 34.7%;
- the CNN baseline from Chapter 4 which reached an accuracy of 39.3%.

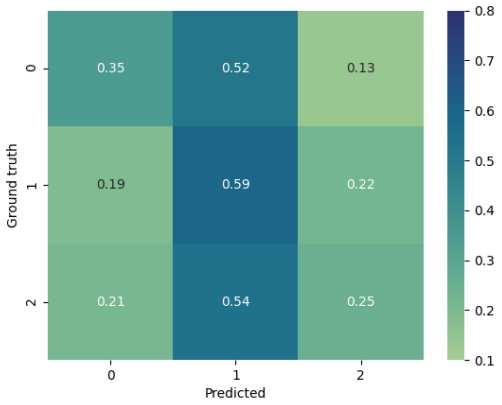

Figure 5.7: Confusion matrix of the CNN tailored to the Shin et al. 2018 dataset of n-back tasks. Classes labelled 0, 1 and 2 represent 0-, 2-, and 3-back respectively.

No significant differences have been found at a 5% threshold, however at a 10% threshold the new dataset-tailored model was found to have greater accuracy than the CNN implemented in Section 5.1 with a p-value of 0.067; indeed that previous model was not significantly better than chance level with an accuracy of 34.7%.

5.3 Discussion

The aim of this chapter was to investigate to what extent tailoring the machine learning models to specific tasks and datasets could affect their performance, and more specifically on the application to n-back level multi-class classification (*RQ4*). It is important to note that while we previously trained the models for each dataset individually, no specific tailoring to one dataset or another was performed. Here the point was to reflect on each task and dataset characteristics in order to adapt the architecture of the models to each specific case, as opposed to just having generic data extraction and model architectures that fit the initial purpose of providing baseline benchmarks on a lot of different datasets.

First of all, when it comes to adapting the baseline models developed for the initial benchmarks to n-back tasks (*RQ4a*), using here the longer epoch durations available (40 seconds as opposed to 10 seconds in the initial benchmarks), we see that improvements are not highlighted. Indeed when taking all the models tested, we ended up having fewer models performing significantly greater than chance level. Again the LDA model remains an interesting choice, because of its simplicity of implementation, and because it is in this case the only model having an evaluation accuracy significantly greater than chance level. We could think that the increased dimensionality of data may have affected negatively the performance, however the dataset-tailored CNN model implemented after the task-tailored baseline used inputs with higher dimensionality and did not suffer from such limitations, presenting performances similar to the initial baseline CNN on 10-second epochs. We can then suspect that patterns in the data useful for a separation between classes may be more pronounced earlier in the task block. This is in line with the fact that the first seconds of a task translate into an activation following an HRF with a peak 4 to 6 seconds after the beginning of the stimulus [26], and that long blocks could make the differentiation of HRFs harder [74].

Furthermore, while investigations from Herff et al. [66] indicated that a longer time window helped increase classification performance, we did not find such a pattern in our investigations (*RQ4b*), quite the opposite in fact. This influence was in most cases not significant and in some cases actually detrimental. This may be related to the fact that in their investigations, they

did not have the same number of examples in the training set with their different time windows due to their sliding window approach, indeed appeared from our investigations in Chapter 4 that a sliding window approach provided an advantage compared to a non-sliding window of the same length, even though limited by the short segment duration. Also, they were interested in within-subject classification as opposed to an unseen subject classification here. It is to be noted that the present findings are coherent with what was found in Chapter 4 where the n-back datasets were the only datasets having negative correlations between window size and accuracy, so this phenomenon may be related to the task itself.

The second experiment consisted in building a deep learning model tailored to a specific dataset, meaning that this dataset was adapted to the fNIRS device used for data acquisition (*RQ4c*). Though the results did not provide significant improvements compared to previous models, its performance was on par with the best classification performances on that specific dataset. This shows us that the higher dimensionality was not limiting in this case, despite the relatively small dataset (702 examples), and that convolutions may have helped extracting useful features for classification.

When diving into the results with the help of the confusion matrix and grand average produced with *BenchNIRS*, we could however see the root of the issue in this classification task. Indeed, the model had difficulties classifying 0- and 3-back with our models, while the classification of 2-back was more reliable as seen in Figure 5.7. We can suspect that 0- and 3-back responses had a similar brain response because they both did not challenge subjects at their best working memory potential: 0-back because the task is too easy, and 3-back because the task is too hard such that subjects may have given up on trying. Indeed, we see in Figure 5.6 that the brain activation is similar for 0- and 3-back. This is supported by Vermeij et al. [165], where lower brain activation for 3-back compared to 2-back is observed, similarly to what we could see in Figure 5.6. They suggested that the task demand could exceed the working memory capacities of subjects. Since subjects, do not necessarily respond the same to different task levels, we could then think in future work of data collection of improved ways to label epochs, for example using a combination of different objective labelling methods such as both task difficulty and performance [111], or a combination of objective and subjective measures such as ISA [83].

5.4 Summary

Once again we demonstrated the utility of *BenchNIRS* to perform the analysis of 6 baseline machine learning models (LDA, SVC, kNN, ANN, CNN and LSTM) specifically for n-back tasks, first with a subject-independent approach (unseen subject classification), and then investigating the influence of the time window size on model performances at classification from fNIRS data on unseen subjects. The results are similar to previous findings with the original baseline benchmarks, highlighting once more how challenging classification from fNIRS data is, and specifically on n-back tasks. Using longer durations of epochs did not help improve performances, and was actually in some cases detrimental to the classification accuracy. This is an indication that the first seconds of a task may be crucial for differentiating n-back levels while the later segments do not help.

Furthermore, *BenchNIRS* was also used to develop a deep learning model (CNN) specifically for one dataset (continuing the validation of *RQ1*). We showed that tailoring this model to the fNIRS device appeared to provide improvement compared to simply using region of interest averaging.

Chapter 6

Exploring transfer learning for fNIRS classification

- *RQ5*: How can using unlabelled segments of the fNIRS recordings help with classification?

6.1 Motivations

In this chapter, which will be the last of experimentation, we will be interested in an overarching limitation that followed us throughout our journey, namely the limited size of the available datasets. This is indeed a potentially significant issue for training machine learning models for classification from fNIRS data especially for deep learning models [59].

A first and obvious solution would be to collect larger fNIRS datasets, however, the field is limited by human and time resources for such enterprises: collecting large datasets requires scientists to spend a lot of time doing data acquisition. Furthermore, fNIRS is far from being prevalent in the industry. All this results in datasets being often of limited size and not always available in open access. For instance, the OpenfNIRS website comports as of 2023 only 14 datasets [1], the 3 biggest datasets comporting recordings from only 43, 24 and 18 subjects.

Another option could be to perform some kind of data augmentation. Such methods have been employed with some success with fNIRS data [116, 117], however here instead of creating artificial data, we will investigate how to make the most out of the real data at disposition. Indeed, in datasets such as the ones we have used so far, segments of the data are annotated with labels that are used for machine learning classification of tasks. However, during the data recording, there might be some cases where data acquisition is not interrupted between tasks, which makes a lot of data in between those annotated segments available. Despite not being labelled, this data remains physiologically relevant: we can assume that the fNIRS device is still properly positioned, hence still recording brain activity. One way to take advantage of these segments is to use them in self-supervised representation learning.

But first, let us take a few steps back and define what we are talking about. In machine learning, we can distinguish different types of learning [23]. Supervised learning is a way of doing machine learning in which each data input has a known output target which is used for training a model, this is what we have used so far for classification in which each input has a label or class. In opposition we have unsupervised learning, in which inputs do not have a known output; in the case of classification for example, each input would be unlabelled because the class to which it belongs is unknown. In between, we have semi-supervised learning, in which the training data is composed of both labelled and unlabelled data.

A way of doing unsupervised learning is with self-supervised learning [81]. In this case, we use the inputs themselves in order to inform the learning process. For example, a part of the

[1] `https://openfnirs.org/data/`

data can be used to learn a representation of itself (this is the principle of autoencoders [59]) or to learn another part of the data. In practice, this type of learning is often used in the context of a pretext task, which will be used to learn a representation of the data in an unsupervised fashion. This pretext task can for example consist in contrastive learning where the aim is to predict whether two inputs are of the same type or not, or in learning spatial or temporal context by splitting the input data into segments and aiming at predicting the segments' temporal or spatial positions. After this representation is learnt, it is used as input for a downstream task, for example a supervised classification task. This is called transfer learning, since the knowledge learnt in the pretext task is then transferred to the downstream task.

While here we will be interested in knowledge transfer using a self-supervised task to take advantage of unlabelled data, transfer learning can also be used for example by pretraining a model on a dataset and using it on another target dataset after fine-tuning for a classification task. This type of approach has been used before for example with EEG on many occasions for BCI applications [173]. It has also been used for fNIRS, where for example Khalil et al. applied transfer knowledge from data of some subjects to others with a CNN [86], which enabled to improve performance and reduce model training time. Dalhoumi et al. [41] also used graph-based transfer learning with fNIRS to transfer knowledge learned on some subjects to a new subject. This type of transfer learning is great when we are interested in an approach halfway between subject-specific and subject-independent: pretraining can be done on prerecorded subjects, and fine-tuning is done on the target subject in order to reduce calibration time of a BCI.

Here we are focusing on subject-independent approaches with no calibration time at all: this is true unseen subject classification. Transfer learning in that case has been proposed before with self-supervised pretext tasks as we have described before for classification from neuroimaging data and can be found in the literature. For example, with EEG Jiang et al. [80] managed to improve the classification accuracy of their models at sleep staging using contrastive learning for a self-supervised pretext task. Banville et al. [8] used, also with EEG, a self-supervised temporal contrastive pretext task, which improved their performance at a sleep scoring task compared to supervised learning. For fNIRS more specifically, Wang et al. [168] for example used a self-supervised pretext task similar to Banville et al. [8] learning temporal context (relative positioning) prior to supervised classification of working memory load, which enabled higher performance compared to supervised approaches alone (SVM and CNN). Also, Ho et al. [68] and Lui et al. [100] used a pretext reconstruction task implementing autoencoders in order to extract features for a downstream mental workload classification task. Such approaches have potential, however, their foundations are only weakly relying on literature knowledge specific to fNIRS and its properties.

In this chapter, we will aim at introducing further knowledge related to the fNIRS properties to design the pretext task: the relationship between HbO and HbR data. Indeed it is now common knowledge that whenever there is brain activation in response to a stimulation, fNIRS signals follow a HRF[26, 99, 176]. Moreover, it is also known that HbO and HbR have a relationship during brain activation which is often described as somewhat negatively correlated [38, 159]. We will therefore use here a pretext task which will consist in the reconstruction of a channel type from the other: reconstructing HbR channels from HbO and vice versa. For this purpose, we will follow up on investigations in Chapter 5. Still focusing on the Shin et al. n-back dataset [155] for reasons stated previously (n-back task as a basis for mental workload estimation, dataset with this task with the largest dataset), we will investigate transfer learning, again using as a basis a CNN type model (more flexibility with deep learning, only model with better performance than chance in initial benchmarking). This will also help make the comparison of our transfer learning approach to previous results more controlled.

This transfer learning approach will be evaluated for task classification from fNIRS data on unseen subjects once again, and this will help us answer the following research questions:

- *RQ5*: How can using unlabelled segments of the fNIRS recordings help with classification?

6.2 Methods

6.2.1 Dataset and signal processing

Once again we here used the same signal processing as in Chapter 5 using *BenchNIRS* [11]. The fNIRS data from the dataset of n-back tasks collected by Shin et al. and published in 2018 [155] was loaded with a sampling rate of 10 Hz. 16 channels for HbO and 16 for HbR from the PFC with a source-detector distance of 30 mm were extracted following Figure 5.3. The motion artefacts were corrected using TDDR and the signal was bandpass filtered (0.01-0.5 Hz). Finally, data was converted into µM.

The epochs from the 26 subjects were extracted for 0-, 2-, and 3-back (234 examples per class) with a duration of 40 seconds from the trigger onset, and using 2 seconds prior to this trigger onset for baseline correction. In addition to that, the signal in between the first and last triggers was scanned in order to extract all possible remaining segments of 42s in the data, resulting in an extra 104 unlabelled epochs that were also corrected based on the 2 first seconds of each segment which were later cropped out. This extraction of extra unlabelled epochs is done because the brain scanner was worn during the entire recording, so we would expect that the signals are of a similar quality as the labelled epochs. No feature extraction was performed so, like before, this resulted in inputs (or examples) of shape 32x400, for which channels are sorted by chromophore types and separated into 2 parts of shape 16x400 each (one part for HbO and one for HbR).

6.2.2 Pretext task: self-supervised representation learning

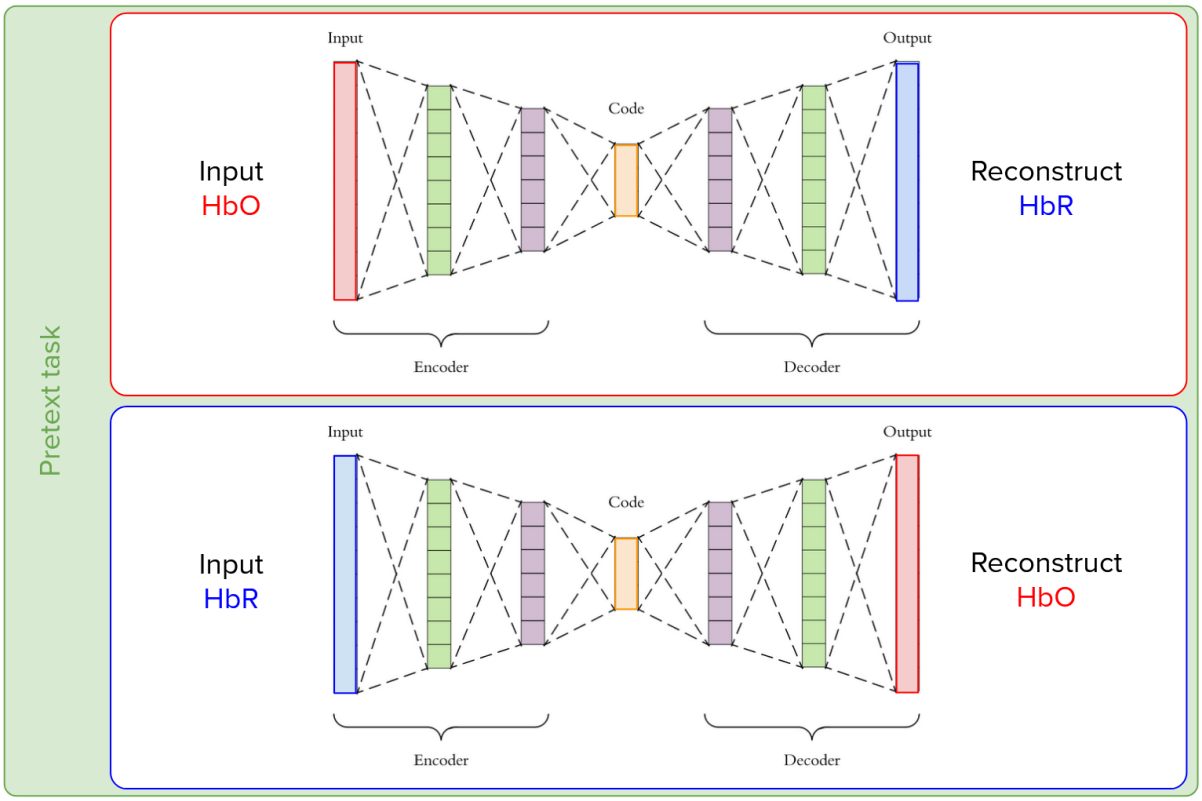

Figure 6.1: Diagram of the concept of representation learning for the pretext task: reconstruction of HbR from HbO, and reconstruction of HbO from HbR.

The pretext task consisted of a self-supervised reconstruction task for representation learning. The goal was to reconstruct HbR from HbO, HbO from HbR. For this purpose, one convolutional encoder-decoder (CED) was implemented for each chromophore type, of which a concept diagram

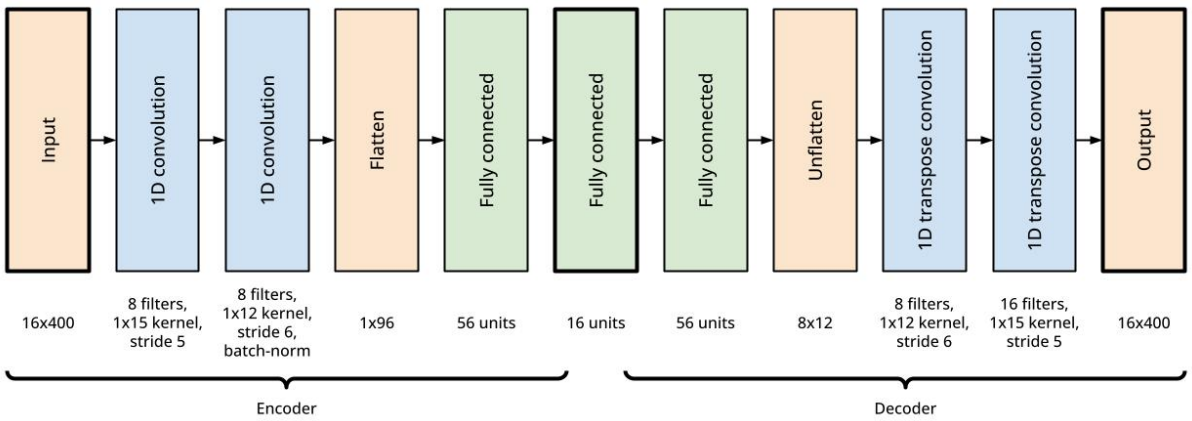

Figure 6.2: CED architecture.

can be found in 6.1. This regression task aims at compressing the information contained in data from a chromophore type with an encoder in such a way that it is possible to decompress it into the other chromophore with a decoder. This process was done for each chromophore with each of the CED. This enabled us to only keep the most useful information in each sample with knowledge of the other matching chromophore, and reduce the dimensionality of the input for the downstream supervised task following up. This acted as feature extraction. To achieve this, each CED had the shape of an hourglass: starting from the initial input size (16x400), reducing it to obtain an encoded version (1x16), and then increasing it back and trying to predict the other chromophore signal (size of 16x400). By training the CEDs on this pretext task, the neural networks learned how to compress the data information of each chromophore in such a way that it contained knowledge about the other chromophore type. As this task consisted of predicting one part of the input from the other part, it did not require any labels, and could then be performed on the full fNIRS recordings including the unlabelled segments.

The architecture of the CED can be found in figure 6.2. First, input data was normalised per channel with min-max scaling using minimums and maximums on the training and validation sets. Each of the two CED (for each chromophore type) was composed of two one-dimensional convolutional layers across the time axis: the first one with 16 input channels and 8 output channels, a kernel size of 15 (one-dimension kernel) and a stride of 5; the second one with 8 input channels and 8 output channels, a kernel size of 12 (one dimension kernel), a stride of 6 and batch normalisation. The resulting matrix was then flattened into a vector which was fed to the fully connected layers of 96, 56 and 16 neurons respectively to result into the encoded version of the data. The decoder followed up with fully connected layers of 56 and 96 neurons followed by a reshaping into a matrix. This was fed into 2 one-dimensional transpose convolutional layers symmetrical to the convolutional layers of the encoder: the first one with 8 input channels and 8 output channels, a kernel size of 12 (one dimension kernel) and a stride of 6; the second one with 16 input channels and 8 output channels, a kernel size of 15 (one dimension kernel) and a stride of 5, reconstructing the data into the output. All the hidden layers activation functions were rectified linear units (ReLU), and we used the Adam optimiser with a mean squared error loss function. The architecture proposed here has been implemented with PyTorch [126].

BenchNIRS [11] was used for a nested group k-fold cross-validation (5-fold for the outer cross-validation and 3-fold for the inner cross-validation) enabling optimisation of hyperparameters and model evaluation of the generalisation capabilities on data from unseen subjects. Input normalisation was performed per channel with min-max scaling using minimums and maximums on the training and validation sets.

Hyperparameter optimisation was performed using *BenchNIRS* on the validation sets with grid search in the same ranges as Chapter 5 [12]:

- learning rate: 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1};
- mini-batch size: 4, 8, 16, 32 and 64.

The validation sets were used to select the model architecture thanks to the training graphs, aiming at a minimal validation mean squared error.

After fine-tuning the model to select the best set of hyperparameters, it was retrained with early stopping similar to the previous section: 20% of validation set randomly left out, 500 epochs maximum, and patience on the validation loss of 5 epochs.

6.2.3 Transfer learning to the downstream task: supervised classification

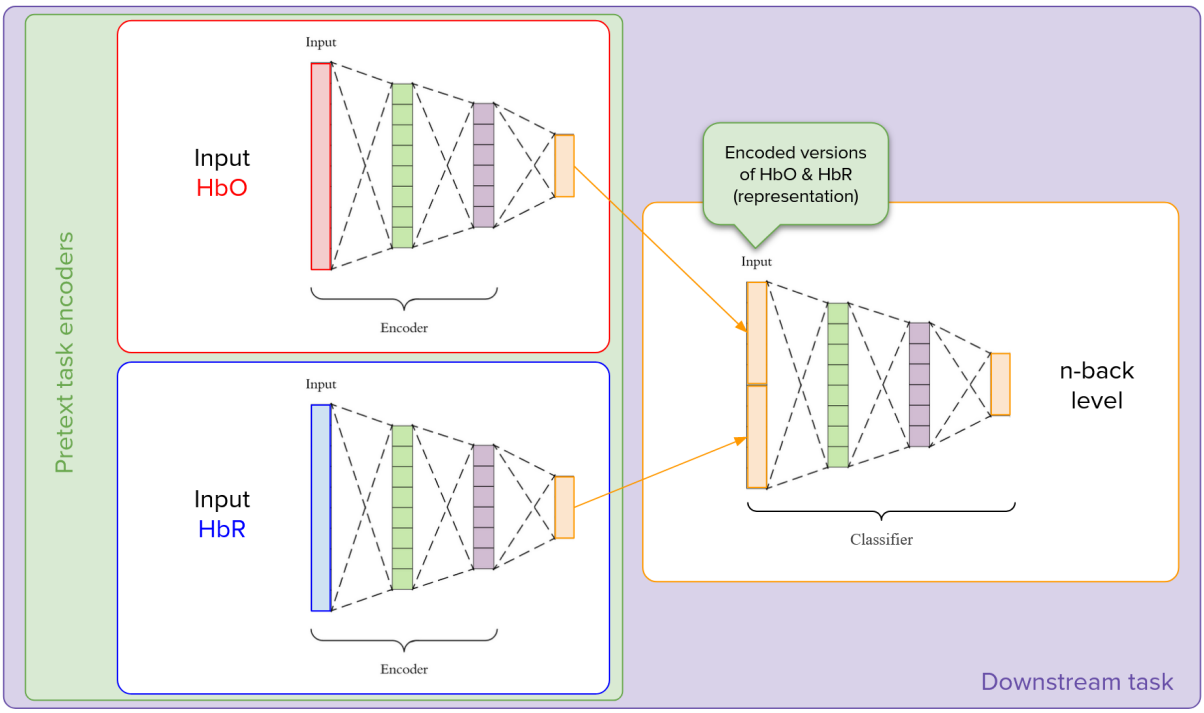

Figure 6.3: Diagram of the concept of transfer learning from the pretext task to the downstream task.

After a regression problem with the pretext task, the downstream task consisted of the initial supervised classification problem of predicting n-back level. This was done with only the labelled epochs. So, after training on the pretext task, the encoder part of the CED was extracted for both HbO to HbR and HbR to HbO. Those two encoders encoded channels for each chromophore, and classification layers were connected to them, such that the outputs of the CED encoders were concatenated and used as input for the classification layers. This process is described in figure 6.3. This means that the resulting overall downstream task model used the whole 32x400 shape data as input to output the class, namely the n-back level (0-, 2-, or 3-back).

The architecture of the model for the downstream task can be found in Figure 6.4. For the downstream task, the connected classification layers used after concatenation of the outputs of the encoders were composed of three fully connected layers: one of 32 neurons and one of 16 neurons with ReLU activation, and the last layer of 3 neurons matching the 3 classes. The cross-entropy loss function was used with the Adam optimiser. For the training on the downstream task, the encoders extracted from the pretext task had all their weights frozen, such that only the classification layer weights were updated here.

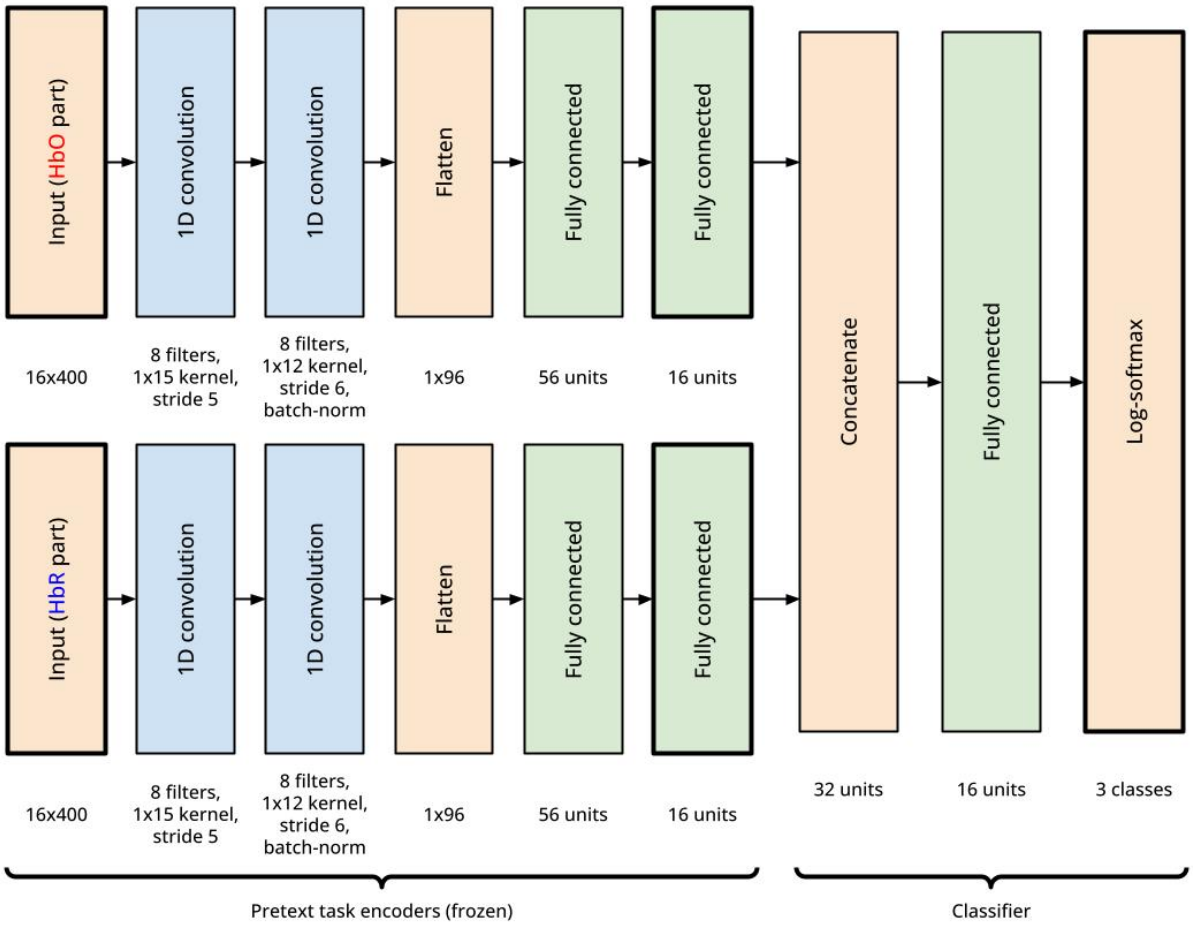

Figure 6.4: Full transfer model architecture.

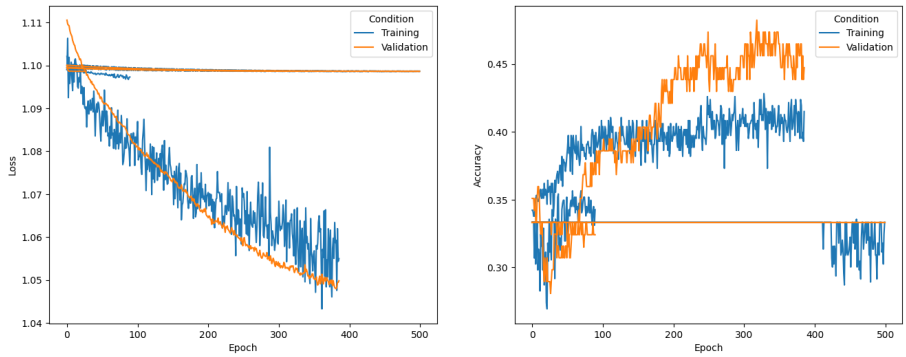

Figure 6.5: Training and validation graphs of the transfer learning approach with both labelled and unlabelled data (cross-entropy loss on the left and accuracy on the right).

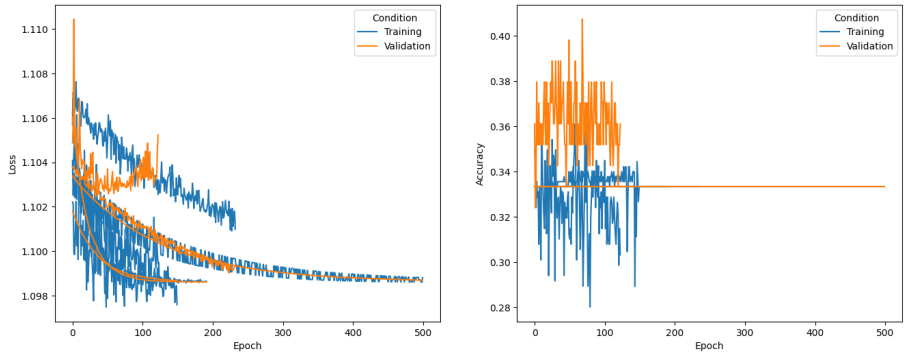

Figure 6.6: Training and validation graphs of the control transfer learning approach with only labelled data (cross-entropy loss on the left and accuracy on the right).

Nested group k-fold cross-validation (5-fold for the outer cross-validation and 3-fold for the inner cross-validation) was once more performed with *BenchNIRS* [11] for hyperparameter optimisation and model evaluation on data from unseen subjects.

Once again, hyperparameter optimisation was performed on the validation sets with grid search in the following ranges [12]::

- the learning rate's values tested for the deep learning models were 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 1×10^{-1};
- the mini-batch sizes tested for the deep learning models were 4, 8, 16, 32, and 64.

The best model architecture was selected using the training graphs, by attempting to have the lowest validation losses possible.

Early stopping was performed for the retrained model the same way as for the pretext task: with 500 epochs maximum and patience of 5 epochs.

6.2.4 Statistics

In order to study the real benefits of using unlabelled segments, two experiments were conducted: first, training the pretext task and downstream task with only labelled segments, which acted as a transfer learning control (training graphs in Figure 6.6); second, training the pretext task with labelled and unlabelled segments, and the downstream task with labelled segments (training graphs in Figure 6.5).

The accuracy of those two trained transfer learning models was compared to chance level using a one-tailed t-test or one-tailed Wilcoxon test (depending on the normality of distribution

of the accuracy on the test sets). Furthermore, we compared the accuracy of the transfer learning approach using both labelled and unlabelled segments to the control transfer learning approach and to the dataset-tailored supervised CNN implemented in Chapter 5 because it was using inputs with the same shape. This was done with one-tailed t-tests or one-tailed Mann-Whitney U (depending on normality and homogeneity of variances verified using Shapiro and Bartlett tests).

6.2.5 Framework extension

All the methods were implemented using *BenchNIRS*. While for the supervised functions, the original library could be used, some new functions have been specifically implemented for self-supervised representation learning.

First of all, this included the implementation in the loading function of a scanning mechanism that would extract all the segments of sufficient length between the labelled epochs.

This included also a new parameter in the data processing function to enable sorting of the channels by chromophore type, which was useful to split the inputs and feed channels from each chromophore to separate representation learning models.

Finally, this included the addition of functions to train regression models (as opposed to only classification initially), and the creation of a new function for implementing the transfer learning approach with nested cross-validation that works with PyTorch models implemented in a modular way. The function takes as arguments 3 model architectures: the encoder architecture, the decoder architecture, and the downstream classifier architecture. It enables the training and hyperparameter tuning of one encoder-decoder model for each chromophore on the pretext task, the extraction of the trained encoders and the transfer to the downstream task for a supervised training with frozen encoder weights, also with fine-tuning. Training graphs for both the pretext task and the downstream task are plotted as well as confusion matrices of the classification evaluation. The model weights and cross-validation configuration are also saved.

The new version of the framework, including the updated library functions and the transfer learning main scrips, is available in the supplementary materials, and the extended documentation is found in the appendices.

6.3 Results

The evaluation on the test sets showed an average accuracy of 34.8% for the transfer learning approach using both labelled and unlabelled data (chance level at 33.3%), and an average accuracy of 33.1% for the control transfer learning approach using only labelled data. These results can be found in Table 6.1. Neither the control transfer learning approach using only labelled segments, nor the transfer learning approach using both labelled and unlabelled data showed accuracy to be significantly greater than chance level (33.3%) at a 5% threshold.

Table 6.1: Evaluation of the transfer learning approach.

Model	**Accuracy**
Transfer (labelled & unlabelled)	0.348 (0.030)%
Transfer (labelled only)	0.331 (0.005)%
Dataset-tailored CNN	0.399 (0.055)%

It appears that the proposed transfer learning approach using the unlabelled segments has improved accuracy at classifying n-back levels compared to the control transfer learning approach

with only labelled segments, however no significant difference has been shown by a Mann-Whitney U test with a significance level of 5% (p-value = 0.173).

However, it appears that the transfer learning implemented here did now help improve the accuracy compared to the dataset-tailored CNN implemented in Chapter 5. It would even appear that the performance is lower, however, no significance has been shown with a Mann-Whitney U test (p-value = 0.098).

6.4 Discussion

In this chapter, we explored the use of unlabelled data for the purpose of classification from fNIRS data (*RQ5*). We developed an approach using knowledge of the fNIRS properties, namely the known relation between HbO and HbR, to learn a representation in an attempt to improve classification performance through transfer learning.

We saw already in Chapter 5 that n-back classification from fNIRS data was a challenge, even when attempting to tailor a model specifically for the task. Such a goal is made difficult because of many limitations, including for example the limited dataset size, but also noisy data and high variability between subjects as highlighted in Figure 5.6 with the large confidence interval bands and also observed in the literature [123].

With our hypothesis that such an approach would improve the performance, the results are unfortunately not very conclusive, with no significant improvement of the classification accuracy in comparison to fully supervised machine learning approaches.

This can be due to the limited augmentation effect of using unlabelled data, indeed when using both labelled and unlabelled data, the unlabelled segments represented only around 13% of the data. However, we could have assumed that learning a representation from the data properties would have been useful regardless. This was unfortunately not the case here.

It is difficult to draw conclusions however, since it is impossible to say with certainty whether the poor results are due to the choice of pretext task, the model type, the model architecture, the data quality or the too-challenging nature of the classification task. Further investigations could be conducted with different options in future work, which is why the framework is open to the community.

It is safe to say however that collecting larger fNIRS n-back dataset would help improve classification with machine learning, or at the very least help have a better understanding of how the brain response is influenced by the various difficulty levels of that task. Furthermore, future work could focus on improving data quality before the pretext task, for example by removing segments for which the relationship between HbO and HbR is too affected by motion artefact despite the correction performed by TDDR.

It could be interesting to explore further *assumption-based pretext tasks* for transfer learning. While here we have designed a pretext task with some domain knowledge, namely the underlying relation between HbO and HbR, further knowledge could be put into the design of such pretext tasks. For example, it could be interesting to use domain knowledge about the HRF for a pretext task. This could be for example using contrastive learning to determine whether the raw signal can or cannot be modelled accurately by a HRF, or using as pretext a reconstruction task of the denoised signals inspired by the autoencoder approach from Gao et al. [55]. Finally, such knowledge-based transfer learning approaches could be combined to domain adaptation [104] from one dataset to another, in order to leverage the combined dataset sizes and eventually also improve generalisation capabilities.

Given the success of such approaches in other fields like computer vision, transfer learning using unsupervised learning has potential, but further investigations are required to prove its full potential.

6.5 Summary

In this chapter, *BenchNIRS* has been extended in order to investigate the interest of transfer learning (further validation of *RQ1*). More specifically, a self-supervised reconstruction pretext task was used to learn a representation of the data incorporating the relation between the two chromophore types HbO and HbR. Knowledge was then transferred to the n-back classification downstream task. This transfer learning approach however did not perform better than supervised learning models, but different pretext tasks introducing domain knowledge could be explored in future work.

Chapter 7

Discussion

Recap of the research questions:

- *RQ1*: How can we make the rigorous development, evaluation, and comparison of machine learning approaches for task classification from fNIRS simpler for researchers?
- *RQ2*: What are the benchmarks of popular machine learning models on various tasks from open access fNIRS datasets?
- *RQ3*: Across these benchmarks, what factors influence the machine learning classification accuracy?
- *RQ4*: How do machine learning approaches tailored to a specific mental workload task compare to baseline benchmarks?
- *RQ5*: How can we take advantage of unlabelled data in fNIRS classification?

7.1 Discussion on the findings

7.1.1 Making machine learning with fNIRS more rigorous and simpler (*RQ1*)

We saw in Chapter 3 how this thesis contributes to making the development, evaluation, and comparison of machine learning models simpler and more rigorous by creating a framework under the form of a software library. This library has key elements that make it more accessible to fNIRS researchers.

First of all, the library is in Python language, which is very high-level and one of the most easily readable programming languages, with a syntax very close to plain English. It benefits from a prodigious community, and has applications, therefore support, in a plethora of scientific fields. This makes it easy for users and contributors to understand and proofread the code, which also contributes to the framework's robustness. More specifically, Python has extensive machine learning and neurophysiology communities which enables the framework to rely on excellent and widely used libraries such as Scikit-learn[1] and PyTorch[2], as well as MNE[3].

Secondly, the library benefits from extensive documentation with a high level of detail. Its style is inspired by the established NumPy Python Style Docstring[4]. This is to make it look familiar for people used to this kind of standardised documentation, but also to make it easily compiled and rendered on a dedicated documentation website generated with Sphinx[5]. This documentation also comports an example use case of how to use the library for developing new

[1] `https://scikit-learn.org/stable/`
[2] `https://pytorch.org/`
[3] `https://mne.tools/stable/index.html`
[4] `https://numpydoc.readthedocs.io/en/latest/format.html`
[5] `https://www.sphinx-doc.org/en/master/`

machine learning models, and scripts used for running the benchmarks are also made available. Instructions on how to install it easily using *pip* as it is hosted on PyPI[6].

Finally, the code has been written to high standards in order to make it easily readable by contributors and the code base has been made as modular as possible to enable easy modifications of the different functions for improvements by external contributors.

7.1.2 Benchmarking machine learning models (*RQ2*)

With the second research question, we were interested in Chapter 4 in presenting results of task prediction from fNIRS data on unseen subjects with 6 baseline models that are popular in BCI. We saw that results were quite different depending on the task: motor execution classification as expected yielded better results than mental tasks, especially compared to n-back for which classifying more reliably than chance level was a challenge. Another takeaway was that deep learning models did not necessarily outperform traditional machine learning models. Overall, performance with this methodology reflecting truly unseen data was found quite lower than in previous work.

We may then ask ourselves the following question: *why is it challenging to reach a performance as high as the existing literature?* It appears that the optimisation of hyperparameters on the test set may unfortunately be a very widespread flaw and source of bias in applied machine learning overall and fNIRS machine learning is therefore not an exception to that. Work by Kapoor & Narayanan [84] reports these issues, and alerts on how widespread they are. The significance of their work gave them the distinction of being named amongst the list of *The 100 Most Influential People in AI* by *Time* under the *Thinkers* category [7].

This type of issue can have regrettable consequences on the fNIRS machine learning community, first of all, because in the long run it can devalue claims in the field, but also because serious researchers may feel discouraged when facing published overstating claims, while being in the incapacity to replicate results themselves. This can also lead work from serious researchers to not being published because allegedly not having groundbreaking enough results to claim novelty.

It should then be emphasised that using a validation set for optimisation is crucial for reflecting unbiased generalisation capabilities of models. One may ask why are we using a validation set for selecting hyperparameters and not models. This is because all the different model performances are planned to be reported from the start of the research, while for hyperparameters, often only the best are reported. In the end, it remains important to consider machine learning as any other data analysis tool. The methodology and what will be reported should be defined rigorously before running any analysis or training any models. It is important to clearly say what models will be tested, with which hyperparameter range, and to report thoroughly all those results in manuscripts, since cherry-picking those reported results is a source of bias, will not represent performance on truly unseen data, and will be misleading for the readers. It is to be noted also that the architecture of a model should only be selected by evaluating on the validation set, either by direct metrics or by looking at training graph curves. Selecting the best architecture based on the test set and only reporting results with the best one will lead to a similar bias to optimisation of hyperparameters on the test set.

To make these concepts clearer we will here illustrate with a hypothetical practical case of how machine learning can be done with poor practices. In this example, the fictional case aims at showing the performance of an LSTM at predicting n-back task levels, using a dataset with labelled examples of 3 different task levels.

- In a first case, data is not split into different resulting in the whole dataset being the training set, and training performance is reported. This is very flawed and would absolutely

[6] `https://pypi.org/project/benchnirs/`

[7] `https://time.com/collection/time100-ai/#thinkers`

not represent generalisations capabilities of a model. Let us call this type of issue *erroneous evaluation*.

- In a second case, a learning rate and a batch size are defined beforehand (or for example using the default hyperparameters from a software), and only those results are reported. As it was planned before investigations, this is not a flawed methodology per se. However the choice of those specific hyperparameters can be questioned, so could be the choice of not optimising them. This may make the results not reflect the best performance possible of that specific model. Let us call this type of issue *genuine unoptimised*.

- In a third case, different learning rates and batch sizes are investigated without methods at random, the best combination is selected by looking at performance on the test set. This is a clear case of flawed methodology leading to biased results. The reason for that is twofold. First, hyperparameters are selected at random, which can be questioned. Second, the test set is used to make an informed decision about hyperparameter selection, which is source of bias and a flawed methodology. Let us call this type of issue *misoptimised leakage*, which is probably the most widespread type of malpractice in machine learning for fNIRS.

- In a forth case, learning rates and batch sizes are investigated with grid search method, but the best combination is selected by looking at the performance on the test set. The test set is used to make an informed decision about hyperparameter selection, which is source of bias and a flawed methodology. Let us call this type of issue *optimised leakage*.

- In a fifth case, examples are segmented using a sliding time window with overlapping, and this process is performed on the whole dataset (including test set) in one go. This approach would result in having some overlapping data points both in the left-out test set as well as the rest of the data, making that test data already seen in the training data, which introduces bias. Let us call this type of issue *overlapping leakage*.

- In a sixth case, min-max scaling normalisation is applied to the data to have all examples between 0 and 1, and this process is performed using maximums and minimums on the whole dataset (including test set). This is source of bias and does not reflect prediction capabilities on unseen data since the test data has been seen to compute the minimums and maximums. Let us call this type of issue *normalisation leakage*.

- In a seventh case, the whole methodology is robust and done according to recommendations in Chapter 3, except that classes are imbalanced due to trial rejection. If most examples end up belonging to a specific class, a dummy classifier classifying this class all the time would appear to have high performance as measured by accuracy, however, it would in fact be a poor classifier as measured by more relevant metrics. Let us call this type of issue *misleading metrics*. Note that to spot this type of issue, it is required to have the number of examples per class after potential trial rejection (because trial rejection may introduce imbalance on an initially balanced dataset).

- In a final case, the methodology is robust except that results are not statistically compared to chance level or state of the art. Claiming any difference of that model with this lack of statistics would be misleading as it could only result from a random effect. Let us call this type of issue *unbacked claims*.

In practice, it is often very challenging to determine what type of issue a paper may present when not enough details are provided to make the reader confident that the methodology is correct at presenting the best possible results (optimised) on unseen data. Hence, if there is suspicion of any flawed methodology being used, or not enough details to ensure it is not flawed, such results should not be trusted.

Finally, it should be noted that randomness can also enter into play when it comes to the model performance, especially for deep learning models which usually use a random state to initialise the weights. The choice of the random state can then lead to outliers that can perform much better or much worse than the average[130]. This is why comparing results with statistical tests is crucial, and evaluating multiple times with different random states is a good practice.

7.1.3 Studying influencing factors (*RQ3*)

For the third research question, we studied also in Chapter 4 the influence of different factors on the classification performance as reflected by the accuracy (since classes were balanced).

More specifically, we saw that the time window of each example used as input of the machine learning models had an influence, and that in most cases, the longer the time window the better the performance.

We saw that the size of the training as little influence on the performance of the models, even though we suspected that the overall maximum available size of the dataset was anyway too small to observe such differences.

We also investigated the use of short sliding windows, which is closer to real-life scenarios, and discovered that the performance was lower than with full time windows but higher than non-sliding short windows. We presume that the performance loss due to the shorter time window may be compensated by the larger amount of data resulting from this method.

Finally, we studied different generalisation goals: predicting task from fNIRS data for unseen subject, as opposed to subject-specific task classification from fNIRS signals (prediction on unseen examples from the same subject as the training). We saw that subject-specific classification could provide an advantage for classification, but such scenarios were not reflecting the reality of BCI applications because the way of evaluating them meant that a BCI based on such approach would require more calibration than actual usage, since there is a lack of evidence that subject-specific approaches can hold their performance across multiple sessions. However, in the case where such approaches were used anyway, one thing to keep in mind would be the training time of the models that would matter in this case since it would have to be done for each new subject after collecting some initial data for calibration. The training time would then be the time to wait between calibration and usage of the BCI, hence traditional machine learning approaches would be a much more cost-efficient choice, especially given the limited performance difference between them and deep learning.

It overall appeared evident once again that the performance was highly dependent on the dataset and task. Multi-class classification was indeed more challenging than binary classification, but specifically, the classification of mental tasks seemed hard compared to motor tasks regardless of modifying different factors (training set size, window size, approach). This would make sense when considering the more complex underlying brain processes involved in mental tasks. The one factor that stood out was the window length, for which longer windows seem to give a real advantage on some tasks that initially were classified decently such as the motor execution. However, increasing the window size did not provide much improvement on mental workload tasks. This would be an element to support the fact that the current machine learning models are limited at classifying such high-level mental processes reliably better than chance.

Another point that is worth mentioning is that machine learning issues such as temporal leakage can appear more on some specific approaches, such as the sliding window approach which classify back to back segments. For example, two successive inputs in that case may be highly related, and should those two successive segments end up in different sets, for example training and test sets, the evaluation of the model's generalisation capabilities would likely be overoptimistic. In our experiments, this was not an issue since group k-fold cross-validation was performed to extract the test sets, therefore making the test sets composed of completely different recordings and subjects from the training and validation sets. However, this is a phenomenon to keep in mind when evaluating models with sliding windows in a within-session configuration,

splitting the same recording between the different sets. This bias risk is less present in other approaches using extracted epochs from the trigger onsets as the different inputs to the models have no direct temporal continuity between each other.

7.1.4 Tailoring machine learning to tasks and datasets (*RQ4*)

The fourth research question, addressed in Chapter 5, was interested in studying how machine learning models could be tailored to a specific mental workload task and to what extent this could influence performance. This was done since it was found in Chapter 4 that model performance was very dependent on the task at hand and the dataset.

We saw that using a longer time window, capturing the full mental workload task, did not necessarily help improve the performance of machine learning models.

In complement of the analysis done for *RQ3* for which we were limited to 10 seconds time windows but saw that the increase of the time window from 2 seconds to 10 seconds was correlated to an improvement of the classification performance and all the datasets except the n-back tasks, we studied with *RQ4* correlation on the n-back tasks with even longer time windows up to 40 seconds. However, this positive correlation was still not observed, even with those longer time windows. This could be task-specific, as n-back tasks indeed require a lot of attention and focus from the participants which can be challenging, this could result in subjects' brain responses being affected by that attention level in addition to the task difficulty, making it difficult to predict reliable information related to task difficulty from the fNIRS data. In order to determine if this lack of correlation beyond 10 seconds for n-back is task-dependent, it could be interesting in future dataset recordings to have longer epochs for example on a motor execution task to see if the correlation of model classification accuracy still correlates to the window size. This would enable us to determine whether reaching a certain window size can become detrimental.

Furthermore, we saw that using all fNIRS channels as input of a deep learning model provided a slight advantage compared to averaging them by regions of interest, proving that higher dimensional inputs can be relevant for classification with deep learning models, especially in the case of models enabling feature extraction like CNNs. It is known that the more input features a deep learning model has, the more chance it has to overfit [59], including overfitting to the hyperparameters if no validation set is used. This highlights the importance of a robust methodology for evaluating machine learning models, even more when the input data has high dimensionality, hence the usefulness of *BenchNIRS* in this case. The example of the development of the CNN on the Shin et al. 2018 n-back dataset in Chapter 5 is indeed yet another validation of the *BenchNIRS* framework and a concrete use case of how it can be useful for researchers to investigate new models. We here used an open-access dataset supported by *BenchNIRS*, however, the framework would also work with private datasets, given that they can be loaded as MNE objects. The framework is here to greatly facilitate the loading and processing of the supported datasets enabling to use of a one-liner Python code for this purpose, but the feature extraction, machine learning methodology, and visualisation can be used for any data loaded in the appropriate format. This results in the framework being relevant for the development of new machine learning models on new datasets.

All these investigations overall reflect however a lasting difficulty to classify n-back levels from fNIRS data, since after all the investigations trying to vary the window size and the number of channels used as inputs in *RQ4*, and the different approaches in *RQ3*, we still did not manage to classify reliably between the different levels of n-back from the fNIRS signals. We can conclude that BCIs aiming at classifying mental workload on even more naturalistic and complex tasks still have a long way to go.

7.1.5 Using unlabelled data with transfer learning (*RQ5*)

Finally, the fifth research question had us focus on the use of unlabelled data with transfer learning. We used a self-supervised pretext task to learn a representation with knowledge on the relationship between HbO and HbR, and then transfer learning for supervised n-back classification.

While the pretext task used domain knowledge about fNIRS properties that were relevant to modelling the relationship between the two chromophore types, this transfer learning approach did not overall enable improvement in the classification of n-back levels compared to fully supervised approaches. This approach was however used as an exploration of unsupervised learning for taking advantage of the significant amount of unlabelled segments in fNIRS recordings in general.

Even though transfer learning, in this case, did not enable performance improvement for n-back classification, transfer learning with pretext representation learning has been proven useful in many fields, primarily computer vision. It can enable improving the performance on a downstream task but also help develop more explainable artificial intelligence (AI) by understanding what the early layers of the model, coming from the pretext task, are achieving.

It is important to keep in mind however that such transfer models can be more time-consuming to train compared to the supervised machine learning approaches since two different machine learning tasks are required. Therefore, they may not be very appropriate for use in the context of within-session classification for BCI, since training is required just before using the system.

7.2 Implications

7.2.1 Making machine learning with fNIRS more rigorous and simpler (*RQ1*)

Answering the first research question led us to build an open source Python software framework for machine learning with fNIRS data, which enables to:

- load and process easily 5 open access datasets of fNIRS recordings;
- perform feature extraction and region of interest averaging;
- prepare fNIRS data with labels for classification;
- use baseline traditional machine learning and deep learning models;
- optimise and evaluate new machine learning models with a state-of-the-art methodology on the open access datasets or new datasets;
- visualise training and evaluation with graphs and other figures;
- compute different metrics for model evaluation that best suit each specific case;
- compute statistics to compare machine learning models, to chance level and to each other;
- vary different factors including signal processing, feature extraction, training set size, window size or sliding window to study the influence on performance.

This framework makes machine learning more accessible for researchers with little machine learning expertise, since it implements the whole methodology preventing flaws and bias, and provides model examples. It also enables making fNIRS more accessible for machine learning specialists with little fNIRS expertise, since it implements state-of-the-art fNIRS pipelines to avoid pitfalls related to the specifics of this neuroimaging technique.

The framework enables researchers using it to save enormous amounts of time when doing machine learning classification with fNIRS, by removing the workload of the data loading, data processing, feature extraction, machine learning methodology, and model evaluation, so that they can focus on one thing: developing novel machine learning models for fNIRS. Of course, despite

machine learning being the point of the pipeline targeted by this framework, it still enables to use various state-of-the-art signal processing techniques and feature extraction methods to the researchers' liking. This is because the quality of the data used as input of machine learning models can be a crucial point for performance, and also that some processing techniques can work better than others in combination with specific machine learning approaches, the basic example being using raw data as opposed to features.

A key implication of the *BenchNIRS* framework is that it enables benchmarking, on shared datasets, state-of-the-art machine learning models that may have initially been developed on restricted datasets. This enables to validate whether claims on a dataset still hold on various open-access datasets.

Furthermore, *BenchNIRS* enables comparisons of models and influencing factors on different datasets, which gives insight into the best approaches to choose for one's specific case. This is useful to enable an informed decision for example for BCI implementation, or automatic classification from fNIRS data in general. We see that this is useful both for academic research but also for use in industry.

A final implication is that it makes machine learning with fNIRS more replicable and more robust at reflecting performance on unseen data. It prevents researchers from making mistakes by adding layers of error catching at the different steps of the pipeline, in a similar way to the Swiss cheese model [142]. This enables to drive the field forward, such that researchers can be confident that advances are validated and trustworthy.

7.2.2 Benchmarking machine learning models and studying influencing factors (*RQ2* & *RQ3*)

In a following part, we provided benchmarks of baseline machine learning models and influencing factors on the classification performance.

This is very useful first of all to validate the usefulness of the framework, by using it to evaluate models with different approaches. This showcases the capabilities of the framework and its different use cases.

Those benchmarks give a starting point for the comparison of models that will encourage researchers to compare more approaches to each other with the same machine learning methodology. This will enable the field to move forward with a solid basis for comparison.

Furthermore, implications are also with regard to the findings themselves, where those machine learning benchmarks had difficulties reaching high performances. Indeed, machine learning is no magical bullet, here to tackle easily all problems with fNIRS. Our findings give indications that classification from fNIRS data is a challenging task, and that it is not ready for reliable industry and commercial applications just yet. It requires a lot more work in investigating what factors are key to machine learning success with fNIRS.

The performance of models proved to be very dependent on the task and dataset at hand, giving important insight that the is no one model to solve all problems, and that the choice should be guided by the task and problem to solve.

However, our findings give early insights into the influence of some factors on classification performance. For example, traditional machine learning and specifically LDA was a decent choice when one wants a simple model that works decently. This is a useful indication for researchers trying to use simple methods for BCI. Also, on some tasks using longer time windows provided a real advantage. Such findings could instruct the study design of future works to maximise the benefits of machine learning for classification from fNIRS data.

7.2.3 Tailoring machine learning to tasks and datasets (*RQ4*)

In answering the fourth research question, we saw that increasing the window size of the inputs of the machine learning models did not necessarily enable an improvement in performance, even

though we were using the whole epochs. This implies that more information is not necessarily beneficial to machine learning classification from fNIRS data. Indeed this gives insight that data selection and feature extraction are quite useful for extracting meaningful information for pattern recognition.

Further investigations with a deep learning model enabling automatic feature extraction on a specific dataset, tailored to the recording device, did enable a decent classification performance compared to the other results. This strengthens the previous findings that each dataset is different and that tailoring a model specifically for a dataset does have its importance.

However, the limited performances on n-back tasks reveal that this task is challenging to classify from fNIRS data alone, possibly because of the difficulty of the task affecting participants' attention or because participants respond differently to those different levels. This implies that results and inferences on such tasks should be taken with caution, and that a complementary way of labelling segments could eventually be used to determine how people are reacting to those different levels of difficulty.

The findings could open the door to alternative ways of improving machine learning performance besides optimising model type or architecture. By taking a step back from machine learning models, it could be interesting to focus also on dataset and labelling quality.

7.2.4 Using unlabelled data with transfer learning (*RQ5*)

For the fifth research question, we explored transfer learning using a self-supervised learning reconstruction pretext task prior to the n-back classification downstream task. This was made to use unlabelled data that is present in every fNIRS recording.

Despite rather inconclusive results in our initial investigations in terms of performance, this could pave the way to exploring more unsupervised pretext tasks. The pretext task used in Chapter 6, reconstructing one chromophore from the other, is one way to introduce domain knowledge into machine learning for classification from fNIRS data so that the model can learn a meaningful representation of it. However, it would be very interesting to study further other tasks to introduce domain knowledge in the learning process, in order for machine learning models to learn a more meaningful representation of the data that encompasses higher-level features reflecting more advanced patterns in the signals. This could even be an opportunity to learn fNIRS representation on many other datasets in an unsupervised way, to then transfer knowledge to a specific downstream task.

7.3 Limitations of the framework

Overall we see the value of *BenchNIRS* in many different cases to explore different machine learning paradigms and approaches. However, it still has some limitations.

First of all, the goal was to focus on the machine learning part of the pipeline. For this purpose, standard state-of-the-art fNIRS processing pipelines have been implemented in the framework, however, more techniques could be added in order to suit the different ways to approach fNIRS data processing in the community, such as more traditional machine learning models like shrinkage LDA [53] or hidden Markov models [140], and more deep learning models like gated recurrent units [31] or transformers [164].

Second, the focus was put on deep learning more specifically, and despite supporting some traditional machine learning models as seen in the various benchmarks, the framework is not currently capable of supporting new models that are not based on neural networks. It would be interesting to study the needs of the community with that regard to determine if the framework should be more flexible with traditional machine learning approaches, or whether it is in fact better to just implement more of the most common traditional machine learning approaches directly into the framework.

Also, the approach to hyperparameter optimisation has been to use grid search for thorough testing of all the combinations in order to select the best. This could potentially be quite time-consuming and other search approaches could be used to save time such as random search [13] or Bayesian optimisation [157]. Furthermore, it is to be noted that instead of using grid search for learning rate fine-tuning, one can use a learning rate scheduler in order to adapt this hyperparameter throughout the epochs. For this kind of design decision on the methodology, it is hard to settle on whether the framework should give the choice to users regarding what machine learning methodology to use, which could result in different methodologies being used making the comparisons more difficult, as opposed to deciding on a specific standard methodology, which makes the comparison easier but could restrict exploration of novel approaches.

Finally, a limitation of the framework is that it is currently a Python repository which requires some technical knowledge of this programming language, and even though it can be considered as a language relatively easy to learn and use as opposed to other ones, it is still not evident to anyone. Efforts were made in that direction by organising workshops for teaching Python for fNIRS, however, more work needs to be done to make it a popular choice in comparison to other tools benefiting from graphical interfaces that can be used without any programming background.

While the *BenchNIRS* framework has initially been created in the context of this thesis, it is designed to be open to community contributions. The goal is that the community makes it its own in order to develop it in the direction it wishes and to suit the needs of most people.

7.4 Future work

We will here suggest future directions for the work done in this thesis which we divide into three categories: short-term for up to 1 year into the future, medium-term for up to 3 years into the future, and long-term for the 10-year outlook.

7.4.1 Short-term

Immediate goals would consist of advertising the framework to start building a user base. This would enable to proof test it and collect feedback from users in order to address immediate limitations that could arise. Besides that, another objective would be to advertise the checklist of recommendations for machine learning with fNIRS and have more people contribute to it to improve it and make it more accessible for any fNIRS researcher. This initiates the effort towards establishing community standards for machine learning with fNIRS. This could be complemented by a short article attempting to describe all the ways machine learning can be flawed and produce biased results with fNIRS data.

Finally, another half would be dedicated to continuing the explorations in transfer learning with different pretext tasks, and publish a comparison of those to supervised machine learning approaches as initiated in Chapter 6. This work would demonstrate further the usefulness of the framework for the implementation of novel machine learning with fNIRS and provide insights into unsupervised learning, little used with fNIRS.

7.4.2 Medium-term

Next, within a 3-year period, the goals would be to continue promoting machine learning standards in the fNIRS community and the *BenchNIRS* framework. The aim would be from an initial user base to establish a contributor base that would drive the machine learning for fNIRS towards properly established standards, and would help to have the community make BenchNIRS its own by contributing to it and improving it to the community's needs.

The goal for *BenchNIRS* would be first to improve the documentation with more examples and guided tutorials, in order to make the framework more accessible. This could include tutorials on

how to use the framework with private datasets. Also, more example models could be implemented such as transformers [164].

Another aspect would also consist of improving the framework by adding support for more datasets, mainly from the OpenfNIRS website's repository, in order to be able to test models on more tasks more easily. In complement to that, the loading of any BIDS dataset with SNIRF files would also be simplified.

Different hyperparameter optimisation algorithms would be supported in addition to grid search, and learning rate scheduling would be added as well.

Furthermore, while *BenchNIRS* currently focuses on machine learning classification. This was initially done because of the discrete nature of labelling in most datasets. However, regression is interesting too, and enables to have more fine-grained predictions. Support for regression would then be made easier to fit every researcher's needs.

Also, *BenchNIRS* currently only supports PyTorch [126] for deep learning, so support for TensorFlow [1], the other popular deep learning library would be added.

Finally, a graphical interface would be added in order to make the framework accessible for researchers with limited programming backgrounds.

7.4.3 Long-term

To finish, on a 10-year time period, the objective would be to have BenchNIRS recognised as an established standard for machine learning with fNIRS amongst the community.

Furthermore, a machine learning fNIRS challenge competition would be organised similarly to other challenges like the *First Passive Brain-Computer Interface Competition on Cross-Session Workload Estimation* [143].

Finally, *BenchNIRS* could become a software suite enabling to help with all the steps of machine learning with fNIRS, from dataset formatting, to machine learning investigations, to model deployment for BCI applications.

7.5 Future directions

Now, when it comes to the future of the field of machine learning with fNIRS, this thesis could lead to a couple of directions.

First, I can see this thesis' work as a useful starting point for establishing a consensus in the community on best practices for machine learning with fNIRS. This could go in the direction of more collaborations between researchers all over the world to agree on a set of best practices that would complement the existing ones. This would lower the barrier of entry for fNIRS researchers wanting to start using machine learning in their research. This direction could be summarised by the following research question:

- *First RQ*: What are the community best practices for machine learning with fNIRS?

Second, I could imagine the framework being used to facilitate the explorations in machine learning for fNIRS classification, which will enable faster and easier transfer of the most recent trends in fundamental machine learning to the fNIRS field. This could ultimately make the neuroimaging field a more significant innovator in fundamental machine learning research as opposed to just being a late adopter of the techniques for its applications. As the interest from the general public in neurotechnologies for everyday life is growing, with more and more neurotechnology companies, this is also a fantastic opportunity to responsibly gather more data with the hope of improving pattern recognition and machine learning for BCIs. This could be summarised into the following research question:

- *Second RQ*: How does reducing the delay between the publication of novel theoretical machine learning models and their adoption in the fNIRS field impact innovation within the field?

Finally, I believe that alternative approaches to supervised learning will become more popular in the community. Some explorations have been started in the thesis by investigating opposite hemoglobin type reconstruction for transfer learning. However, the playground is left open with this thesis' work and I believe such approaches are only at their early stages in the field of fNIRS. Making machine learning more accessible through a framework and better standards will enable talented experts of subfields of fNIRS to get started with confidence with machine learning, injecting their innovation into the algorithms which may in the end disrupt the world of fNIRS BCIs. This could be turned into the following research question:

- *Third RQ*: How does making machine learning more accessible to fNIRS researchers impact innovation in BCIs?

While those three research questions are relatively open, they could shape a long-term vision for machine learning with fNIRS research.

Chapter 8

Conclusions

8.1 Summary of the contributions

To sum up, the contributions of this thesis can be divided into two main parts.

A first core contribution is a framework for machine learning with fNIRS called *BenchNIRS*. This framework is an open-source Python software and enables to:

- load open-access fNIRS datasets from the community;
- process fNIRS signal with state-of-the-art methods;
- extract features;
- use a state-of-the-art machine learning methodology to develop new machine learning models including training, optimisation, evaluation, and comparison of models for classification from fNIRS data;
- produce clear visualisations for monitoring training and presenting results.

BenchNIRS enables the use of different approaches and paradigms. This includes the ability to use both supervised and self-supervised learning and to perform transfer learning. It also gives the ability to study different influencing factors on the performance of the machine learning models (time window length, number of training examples, sliding window) with different generalisation goals (subject-independent or subject-specific). It comes with 6 machine learning models pre-implemented, including LDA, SVC, kNN, ANN, CNN and LSTM, and provides example scripts for the evaluation of machine learning models with statistical analysis. This framework of more than 3,500 lines of code enables to simplify machine learning for fNIRS and makes it much less time-consuming to develop new models by enabling the implementation of a whole pipeline in a dozen lines of code. It makes machine learning accessible for fNIRS researchers with little machine learning experience, and makes fNIRS accessible for machine learning researchers with little fNIRS experience, bridging the gap between those two advanced and technical fields.

The second core contribution consists of providing benchmarks of different machine learning approaches for classification from fNIRS on various tasks. This includes:

- the evaluation of various baseline machine learning models including LDA, SVC, kNN, ANN, CNN and LSTM;
- the evaluation of the influence of different factors on the machine learning performance including the size of the time window, the number of training examples, and the use of a sliding window.
- the evaluation of the influence of different generalisation goals including subject-independent classification and subject-specific classification;

- the evaluation of the tailoring of machine learning models on specific tasks and datasets;
- the evaluation of different machine learning paradigms including supervised learning, self-supervised learning, and transfer learning.

This contributes to giving a starting point for comparison of many different machine learning approaches with fNIRS. It also provides indications as to what models and approaches are best suited for each case, which is useful when it comes to choosing an algorithm for BCI application.

8.2 Summary of the findings

8.2.1 Building a framework for machine learning with fNIRS

In answering *RQ1* we built an open-source Python framework for facilitating research in machine learning with fNIRS. We described the development in multiple steps, first with the signal processing library *NIRSimple*, second with the initial *BenchNIRS* framework enabling the comparison of different supervised machine learning classification approaches with fNIRS data which was validated by using it to produce benchmarks, third by extending it to self-supervised representation learning and transfer learning which was validated with a deep learning example on a n-back dataset. We saw throughout the thesis how this framework was used to make the comparison more accessible by providing an easy-to-use API, and examples with baseline machine learning comparisons, all available in open source online[1], and open to contributions from the community for improvements and moving towards more robust and standardised machine learning for fNIRS.

8.2.2 Benchmarking machine learning models and studying influencing factors

The *BenchNIRS* framework was used to answer *RQ2* (benchmarks of baseline machine learning models for classification with fNIRS) and *RQ3* (factors influencing the performance of machine learning with fNIRS).

In analysing 6 baseline machine learning models (LDA, SVC, kNN, ANN, CNN and LSTM), we found that none of them consistently stood out at task classification from fNIRS data except the LDA, which despite being simple was reliably classifying task with an accuracy significantly better than chance on all the datasets and was outperforming significantly other models in many cases.

We then studied the influence of different factors on the models' performances. We saw that at the scale of the datasets studied, the number of training examples did not influence the classification accuracy of models. We found that most models that had a classification accuracy significantly greater than chance level increased their accuracy as the time window length increased. Also, we saw that using a small sliding window did not improve performance compared to using longer non-sliding windows. Finally, we found that, as expected, subject-specific classification enabled to classify more reliably tasks from fNIRS data compared to unseen subject classification, though this approach has practical limitations for BCI applications.

All in all, all those benchmarks highlighted that with the robust methodology provided by *BenchNIRS* ensuring unbiased evaluation, results were consequently lower than some of the existing literature, indicating that more research is required in order to make fNIRS BCIs more reliable.

[1] `https://gitlab.com/HanBnrd/benchnirs`

8.2.3 Tailoring machine learning to tasks and datasets

RQ4 had us interested in tailoring machine learning models to n-back tasks, and also more specifically one dataset. Once again, the usefulness of *BenchNIRS* was highlighted for that study and enabled the production of further results with baseline machine learning models.

We highlighted that extending further the window length from 10 seconds to 40 seconds did not help improve the classification accuracy of the tested models. Furthermore, no correlations of the model accuracy to the window length were found in this new study on n-back tasks.

Furthermore, we developed a deep learning CNN model specifically for one of the n-back datasets and adapted its architecture to the task duration and the fNIRS device at hand. This appeared to have the potential to improve the accuracy of the model compared to a similar architecture using region of interest averaging. However, it did not improve the performance compared to the best initial baseline models using only 10-second time windows.

We concluded that n-back classification remains a challenging goal, and suggest ways this challenge could be overcome, for example by using different ways to label the data.

8.2.4 Using unlabelled data with transfer learning

The final findings were related to the use of unlabelled segments of the data in order to perform representation learning and transfer learning.

While we explored here an interesting concept with pretext tasks to learn knowledge then transferred to a downstream classification task, we did not observe an improved classification accuracy with this approach compared to initial supervised baseline machine learning models.

We took those insights to then suggest future work tracks aiming at performing more knowledge-based machine learning with fNIRS, and that could benefit from using unlabelled segments of fNIRS recordings that can usually be found in a non-negligible amount in typical datasets. Such domain knowledge-based machine learning could be performed by taking advantage of the complex properties of the fNIRS data measuring both HbO and HbR, but also of the hemodynamic specificities of the brain responses.

8.3 Closing remarks

To conclude this work with a few words, I wanted to share some overall thoughts about machine learning for fNIRS.

8.3.1 How to choose the best machine learning model for fNIRS?

First, some considerations about identifying the best-performing machine learning model for fNIRS data classification. This remains a challenging task. Despite this thesis bringing elements to the table for answering this question, the main answer appears to be: first of all, classifying tasks from fNIRS data is difficult, and classifying higher-level cognitive states or processes would surely be even more difficult. Nonetheless, some hints are given in this work. Traditional machine learning should not be ashamed, as some of its models often perform better than more advanced deep learning approaches. More specifically LDA with carefully chosen features, despite being very simple in its implementation, optimisation, and computing power demand, kept surprising us by almost always beating other models. This highlights the great importance of expert knowledge when doing machine learning with fNIRS, as the extraction of relevant features can play a key role in the successful implementation of models. Another highlight in the deep learning category would be CNN, with a rather efficient way to extract features automatically, enabling to work on fNIRS data without manual feature extraction. It however appears to me that further research should be conducted with such models in order to address evident issues of small dataset sizes. Furthermore, we have seen that differences between different models are far from huge, so I then

want to emphasise the fact that there is most likely no model that is reliably better than others at classifying fNIRS data, and that existing claims of models outperforming others may be explained rather by the optimisation and the discovery of the sweet spot when it comes to adapting the architecture to a task or a dataset.

Beyond identifying the best performing model for classification from fNIRS data, some thought should also be put into what model is best suited in a specific case. First of all, models perform differently depending on the task at hand, and the dataset. Secondly, other constraints could affect the choice of a model for an application. Indeed, considerations like computing demand, development, and implementation time, should also come into play. For example, developing and optimising deep learning models can be tedious and time-consuming, in addition to requiring consequent computing resources to train. If furthermore, the performance of models depends on each specific use case, someone wanting to develop quick algorithms performing decently may well be interested in traditional machine learning approaches.

8.3.2 How to make machine learning with fNIRS better?

Now, a question with an optimistic future in mind: how do we make machine learning with fNIRS better? Here again, this work provides elements to help achieve that goal.

First of all, in my opinion, it starts by setting a solid basis, validating that basis, and reproducing it. Only then can we build up. This needs to be also supported by more research into the explainability of machine learning, to uncover the reasons why some approaches work better than others. Second, it continues with the community gathering, sharing, and helping each other in an honest but non-judgemental way. Only then can ideas stem and develop. Finally, I believe it is achieved with explorations in order to find better machine learning approaches, better paradigms, and better ideas. The framework produced in this work is designed to have researchers spend less time on the steps that need to be done in order to produce reliable research, and redirect that time and energy to what in my opinion matters most: exploring. Not just exploring machine learning only, or fNIRS only, but both in a symbiotic relationship in order to make those two scientific tools work best for each other. And while this framework provides guidance for the development of machine learning with fNIRS it is bound to be collaborative, open to contributions to extend it and make it more flexible to other validated approaches, in order for it to not limit innovation.


BENCH
NIRS

01010000 01100001 01101111 01101100 01100001 00100000 01001010 01100001 01101101
01101001 01100101 00001010 01001101 01100001 01111000 00100000 01001101 01101001
01100011 01101000 01100101 01101100 00100000 01001010 01100101 01110010 01100101
01101101 01101001 01100101 00001010 01000001 01101110 01101110 01100101 00100000
01001101 01100001 01110101 01110010 01100101 01100101 01101110 00001010 01000001
01100001 01110010 01101111 01101110 00100000 01000001 01100100 01110010 01101001
01100001 01101110 00100000 01000001 01101100 01100001 01101110 00100000 01000001
01101100 01100101 01101011 01110011 01100001 01101110 01100100 01110010 01100001
00100000 01000001 01101100 01111001 01110011 01100101 01100101 00100000 01000001
01101110 01100100 01110010 01101001 01100001 01101110 01100001 00100000 01000010
01110010 01100001 01101110 01100100 01101111 01101110 00100000 01000011 01100001
01101100 01101100 01110101 01101101 00100000 01000011 01101111 01101101 01100101
00100000 01000011 01110010 01100001 01101001 01100111 00100000 01000100 01101001
01101101 01101001 01110100 01110010 01101001 01101111 01110011 00100000 01000101
01100100 01100111 01100001 01110010 00100000 01000101 01101001 01101011 01100101
00100000 01000101 01101100 01101001 01100110 00100000 01000101 01101101 01110010
01111001 01110011 00100000 01000111 01101001 01101111 01110110 01100001 01101110
01101110 01101001 00100000 01000111 01101001 01110011 01100101 01101100 01100001
00100000 01000111 01110101 01110011 01110100 01100001 01110110 01101111 00100000
01001000 01101111 01110010 01101001 01100001 00100000 01001000 01110101 01101001
01101101 01101001 01101110 00100000 01001001 01101111 01100001 01101110 01101110
01100001 00100000 01001010 01101111 01110011 01101000 00100000 01001010 01101111
01111001 00100000 01001010 01110101 01100001 01101110 00100000 01001010 01110101
01110011 01110100 01101001 01101110 01101001 01100101 01101110 00100000 01001010
01110111 01100001 01101110 00100000 01001011 01100101 01100101 01110010 01110100
01101000 01111001 00100000 01001011 01100101 01110110 01101001 01101110 00100000
01001011 01101000 01100001 01101001 01110010 01101001 01100100 01101001 01101110
01100101 00100000 01001101 01100001 01101110 01101001 00100000 01001101 01100101
01101100 01101001 01110011 01110011 01100001 00100000 01001101 01101001 01101110
01100001 00100000 01010000 01100001 01100010 01101100 01101111 00100000 01010000
01100101 01110000 01101001 01110100 01100001 00100000 01010010 01101111 01100010
01100101 01110010 01110100 00100000 01010011 01100001 01110010 01100001 00100000
01010011 01100101 01110010 01100101 01101110 01100001 00001010 01010010 01100001
01110000 01101000 01100001 01100101 01101100 00100000 01001101 01100001 01101101
01100001 01101110 00100000 01010000 01100001 01110000 01100001 00100000 01001101
01100101 01101101 01100101 00100000 01010000 01100101 01110000 01100101 00001010
01001011 01100101 01110010 01100001 01101110

Bibliography

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, software available from tensorflow. org (2015). *URL https://www. tensorflow. org*, 2015.

[2] Daniel Afergan, Evan M Peck, Erin T Solovey, Andrew Jenkins, Samuel W Hincks, Eli T Brown, Remco Chang, and Robert JK Jacob. Dynamic difficulty using brain metrics of workload. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3797–3806, 2014.

[3] Haleh Aghajani, Marc Garbey, and Ahmet Omurtag. Measuring mental workload with eeg+ fnirs. *Frontiers in human neuroscience*, 11:359, 2017.

[4] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] Aurélien Appriou, Andrzej Cichocki, and Fabien Lotte. Towards robust neuroadaptive hci: exploring modern machine learning methods to estimate mental workload from eeg signals. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2018.

[6] Brian P Bailey, Joseph A Konstan, and John V Carlis. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *Interact*, volume 1, pages 593–601, 2001.

[7] SuJin Bak, Jinwoo Park, Jaeyoung Shin, and Jichai Jeong. Open-access fnirs dataset for classification of unilateral finger-and foot-tapping. *Electronics*, 8(12):1486, 2019.

[8] Hubert Banville, Isabela Albuquerque, Aapo Hyvärinen, Graeme Moffat, Denis-Alexander Engemann, and Alexandre Gramfort. Self-supervised representation learning from electroencephalography signals. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

[9] Jackson Beatty. Task-evoked pupillary responses, processing load, and the structure of processing resources. *Psychological bulletin*, 91(2):276, 1982.

[10] Johann Benerradi, Horia A. Maior, Adrian Marinescu, Jeremie Clos, and Max L. Wilson. Exploring machine learning approaches for classifying mental workload using fnirs data from hci tasks. In *Proceedings of the Halfway to the Future Symposium 2019*, pages 1–11, 2019.

[11] Johann Benerradi, Jeremie Clos, Aleksandra Landowska, Michel F Valstar, and Max L Wilson. Benchmarking framework for machine learning classification from fnirs data. *Frontiers in Neuroergonomics*, 4:994969, 2023.

[12] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478, 2012.

[13] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[14] Sagarika Bhattacharjee, Rajan Kashyap, Turki Abualait, Shen-Hsing Annabel Chen, Woo-Kyoung Yoo, and Shahid Bashir. The role of primary motor cortex: more than movement execution. *Journal of motor behavior*, 53(2):258–274, 2021.

[15] Christopher M Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

[16] David Boas and Maria Angela Franceschini. Near infrared imaging. `http://www.scholarpedia.org/article/Near_infrared_imaging`, 2009. Accessed: 2020-05.

[17] David A Boas, Anders M Dale, and Maria Angela Franceschini. Diffuse optical imaging of brain activation: approaches to optimizing image sensitivity, resolution, and accuracy. *Neuroimage*, 23:S275–S288, 2004.

[18] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[19] Sabrina Brigadoi, Lisa Ceccherini, Simone Cutini, Fabio Scarpa, Pietro Scatturin, Juliette Selb, Louis Gagnon, David A Boas, and Robert J Cooper. Motion artifacts in functional near-infrared spectroscopy: a comparison of motion correction techniques applied to real cognitive data. *Neuroimage*, 85:181–191, 2014.

[20] Sabrina Brigadoi and Robert J Cooper. How short is short? optimum source–detector distance for short-separation channels in functional near-infrared spectroscopy. *Neurophotonics*, 2(2):025005, 2015.

[21] Karel A Brookhuis and Dick de Waard. Assessment of drivers'workload: Performance and subjective and physiological indexes. *Stress, workload and fatigue*, 2001.

[22] Scott C Bunce, Meltem Izzetoglu, Kurtulus Izzetoglu, Banu Onaral, and Kambiz Pourrezaei. Functional near-infrared spectroscopy. *IEEE engineering in medicine and biology magazine*, 25(4):54–62, 2006.

[23] Andriy Burkov. *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, QC, Canada, 2019.

[24] Shannon M Burns. Ucla fnirs bootcamp. `https://www.youtube.com/watch?v=TEMNe5ROsw4&list=PLJqHk_LwA-fctJUP4zFzmL0cHCrzxz0MQ`, 2019. Accessed: 2019-02.

[25] Stephen Butterworth et al. On the theory of filter amplifiers. *Wireless Engineer*, 7(6):536–541, 1930.

[26] Richard B Buxton, Kâmil Uludağ, David J Dubowitz, and Thomas T Liu. Modeling the hemodynamic response to brain activation. *Neuroimage*, 23:S220–S233, 2004.

[27] Murat Perit Çakır, Murat Vural, S"uleyman "Ozg"ur Koç, and Ahmet Toktaş. Real-time monitoring of cognitive workload of airline pilots in a flight simulator with fnir optical brain imaging technology. In *International Conference on Augmented Cognition*, pages 147–158. Springer, 2016.

[28] Daniel Carius, Lisa Hörnig, Patrick Ragert, and Elisabeth Kaminski. Characterizing cortical hemodynamic changes during climbing and its relation to climbing expertise. *Neuroscience letters*, 715:134604, 2020.

[29] Mickael Causse, Zarrin Chua, Vsevolod Peysakhovich, Natalia Del Campo, and Nadine Matton. Mental workload and neural efficiency quantified in the prefrontal cortex using fnirs. *Scientific reports*, 7(1):1–15, 2017.

[30] Justin Chan, Sarah Power, and Tom Chau. Investigating the need for modelling temporal dependencies in a brain-computer interface with real-time feedback based on near infrared spectra. *Journal of Near Infrared Spectroscopy*, 20(1):107–116, 2012.

[31] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[32] Maureen Clerc, Laurent Bougrain, and Fabien Lotte. *Brain-Computer Interfaces 1*. Wiley-ISTE, 2016.

[33] Fokie Cnossen, Theo Meijman, and Talib Rothengatter. Adaptive strategy changes as a function of task demands: a study of car drivers. *Ergonomics*, 47(2):218–236, 2004.

[34] Patricia Cohen, Stephen G West, and Leona S Aiken. *Applied multiple regression/correlation analysis for the behavioral sciences*. Psychology press, 2014.

[35] Robert Cooper, Juliette Selb, Louis Gagnon, Dorte Phillip, Henrik W Schytz, Helle K Iversen, Messoud Ashina, and David A Boas. A systematic comparison of motion artifact correction techniques for functional near-infrared spectroscopy. *Frontiers in neuroscience*, 6:147, 2012.

[36] Mark Cope. The application of near infrared spectroscopy to non invasive monitoring of cerebral oxygenation in the newborn infant. *Department of Medical Physics and Bioengineering*, 342, 1991.

[37] Xu Cui, Signe Bray, Daniel M Bryant, Gary H Glover, and Allan L Reiss. A quantitative comparison of nirs and fmri across multiple cognitive tasks. *Neuroimage*, 54(4):2808–2821, 2011.

[38] Xu Cui, Signe Bray, and Allan L Reiss. Functional near infrared spectroscopy (nirs) signal improvement based on negative correlation between oxygenated and deoxygenated hemoglobin dynamics. *Neuroimage*, 49(4):3039–3046, 2010.

[39] Xu Cui, Signe Bray, and Allan L Reiss. Speeded near infrared spectroscopy (nirs) response detection. *PLoS one*, 5(11), 2010.

[40] Edward Cutrell, Mary Czerwinski, and Eric Horvitz. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. In *Human-Computer Interaction: INTERACT*, volume 1, page 263, 2001.

[41] Sami Dalhoumi, Gérard Derosiere, Gérard Dray, Jacky Montmain, and Stéphane Perrey. Graph-based transfer learning for managing brain signals variability in nirs-based bcis. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems: 15th International Conference, IPMU 2014, Montpellier, France, July 15-19, 2014, Proceedings, Part II 15*, pages 294–303. Springer, 2014.

[42] Patrick W Dans, Stevie D Foglia, and Aimee J Nelson. Data processing in functional near-infrared spectroscopy (fnirs) motor control research. *Brain Sciences*, 11(5):606, 2021.

[43] T.M. De Silva and F.M. Faraci. Chapter 31 - hypertension. In Louis R. Caplan, José Biller, Megan C. Leary, Eng H. Lo, Ajith J. Thomas, Midori Yenari, and John H. Zhang, editors, *Primer on Cerebrovascular Diseases (Second Edition)*, pages 153–157. Academic Press, San Diego, second edition edition, 2017.

[44] David T Delpy, Mark Cope, Pieter van der Zee, SR Arridge, Susan Wray, and JS Wyatt. Estimation of optical pathlength through tissue from direct time of flight measurement. *Physics in Medicine & Biology*, 33(12):1433, 1988.

[45] Condell Eastmond, Aseem Subedi, Suvranu De, and Xavier Intes. Deep learning in fnirs: a review. *Neurophotonics*, 9(4):041411–041411, 2022.

[46] F Thomas Eggemeier, Glenn F Wilson, Arthur F Kramer, and Diane L Damos. Workload assessment in multi-task environments. *Multiple-task performance*, pages 207–216, 1991.

[47] Ahmed Faress and Tom Chau. Towards a multimodal brain–computer interface: combining fnirs and ftcd measurements to enable higher classification accuracy. *Neuroimage*, 77:186–194, 2013.

[48] Marco Ferrari and Valentina Quaresima. A brief review on the history of human functional near-infrared spectroscopy (fnirs) development and fields of application. *Neuroimage*, 63(2):921–935, 2012.

[49] Frank A Fishburn, Ruth S Ludlum, Chandan J Vaidya, and Andrei V Medvedev. Temporal derivative distribution repair (tddr): a motion correction method for fnirs. *Neuroimage*, 184:171–179, 2019.

[50] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[51] Hannah J Foy, Patrick Runham, and Peter Chapman. Prefrontal cortex activation and young driver behaviour: a fnirs study. *PLoS one*, 11(5):e0156512, 2016.

[52] Lex Fridman, Bryan Reimer, Bruce Mehler, and William T Freeman. Cognitive load estimation in the wild. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–9, 2018.

[53] Jerome H Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[54] Naomi P Friedman and Trevor W Robbins. The role of prefrontal cortex in cognitive control and executive function. *Neuropsychopharmacology*, pages 1–18, 2021.

[55] Yuanyuan Gao, Hanqing Chao, Lora Cavuoto, Pingkun Yan, Uwe Kruger, Jack E Norfleet, Basiel A Makled, Steven Schwaitzberg, Suvranu De, and Xavier Intes. Deep learning-based motion artifact removal in functional near-infrared spectroscopy. *Neurophotonics*, 9(4):041406–041406, 2022.

[56] Audrey Girouard, Erin Treacy Solovey, Leanne M Hirshfield, Evan M Peck, Krysta Chauncey, Angelo Sassaroli, Sergio Fantini, and Robert JK Jacob. From brain signals to adaptive interfaces: using fnirs in hci. In *Brain-Computer Interfaces*, pages 221–237. Springer, 2010.

[57] Audrey Girouard, Erin Treacy Solovey, and Robert JK Jacob. Designing a passive brain computer interface using real time classification of functional near-infrared spectroscopy. *International Journal of Autonomous and Adaptive Communications Systems*, 6(1):26–44, 2013.

[58] Florens Goldbeck, Alina Haipt, David Rosenbaum, Tim Rohe, Andreas J Fallgatter, Martin Hautzinger, and Ann-Christine Ehlis. The positive brain–resting state functional connectivity in highly vital and flourishing individuals. *Frontiers in Human Neuroscience*, 12:540, 2019.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[60] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, et al. Meg and eeg data analysis with mne-python. *Frontiers in neuroscience*, 7:267, 2013.

[61] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013.

[62] Eija Haapalainen, SeungJun Kim, Jodi F Forlizzi, and Anind K Dey. Psycho-physiological measures for assessing cognitive load. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 301–310, 2010.

[63] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[64] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[65] Johannes Hennrich, Christian Herff, Dominic Heger, and Tanja Schultz. Investigating deep learning for fnirs based bci. In *2015 37th Annual international conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2844–2847. IEEE, 2015.

[66] Christian Herff, Dominic Heger, Ole Fortmann, Johannes Hennrich, Felix Putze, and Tanja Schultz. Mental workload during n-back task—quantified in the prefrontal cortex using fnirs. *Frontiers in human neuroscience*, 7:935, 2014.

[67] Christian Herff, Dominic Heger, Felix Putze, Johannes Hennrich, Ole Fortmann, and Tanja Schultz. Classification of mental tasks in the prefrontal cortex using fnirs. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2160–2163. IEEE, 2013.

[68] Thi Kieu Khanh Ho, Jeonghwan Gwak, Chang Min Park, and Jong-In Song. Discrimination of mental workload levels from multi-channel fnirs using deep leaning-based approaches. *Ieee Access*, 7:24392–24403, 2019.

[69] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[70] Keum-Shik Hong, M Jawad Khan, and Melissa J Hong. Feature extraction and classification methods for hybrid fnirs-eeg brain-computer interfaces. *Frontiers in human neuroscience*, 12:246, 2018.

[71] Keum-Shik Hong, Noman Naseer, and Yun-Hee Kim. Classification of prefrontal and motor cortex signals for three-class fnirs–bci. *Neuroscience letters*, 587:87–92, 2015.

[72] Yoko Hoshi, Brian H Tsou, Vincent A Billock, Masato Tanosaki, Yoshinobu Iguchi, Miho Shimada, Toshikazu Shinba, Yoshifumi Yamada, and Ichiro Oda. Spatiotemporal characteristics of hemodynamic changes in the human lateral prefrontal cortex during working memory tasks. *Neuroimage*, 20(3):1493–1504, 2003.

[73] Zhe Huang, Liang Wang, Giles Blaney, Christopher Slaughter, Devon McKeon, Ziyu Zhou, Robert J. K. Jacob, and Michael C. Hughes. The tufts fnirs mental workload dataset & benchmark for brain-computer interfaces that generalize. In *Proceedings of the Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2021.

[74] S.A. Huettel, A.W. Song, and G. McCarthy. *Functional Magnetic Resonance Imaging*. Sinauer, 2014.

[75] Theodore J Huppert, Maria Angela Franceschini, and David A Boas. Noninvasive imaging of cerebral activation with diffuse optical tomography. In *In Vivo Optical Imaging of Brain Function. 2nd edition*. CRC Press/Taylor & Francis, 2009.

[76] Theodore J Huppert, Richard D Hoge, Solomon Gilbert Diamond, Maria Angela Franceschini, and David A Boas. A temporal comparison of bold, asl, and nirs hemodynamic responses to motor stimuli in adult humans. *Neuroimage*, 29(2):368–382, 2006.

[77] Shamsi T Iqbal, Xianjun Sam Zheng, and Brian P Bailey. Task-evoked pupillary response to mental workload in human-computer interaction. In *CHI'04 extended abstracts on Human factors in computing systems*, pages 1477–1480, 2004.

[78] Sahar Jahani, Seyed K Setarehdan, David A Boas, and Meryem A Yücel. Motion artifact detection and correction in functional near-infrared spectroscopy: a new hybrid method based on spline interpolation method and savitzky–golay filtering. *Neurophotonics*, 5(1):015003–015003, 2018.

[79] Vinay Jayaram and Alexandre Barachant. Moabb: trustworthy algorithm benchmarking for bcis. *Journal of neural engineering*, 15(6):066011, 2018.

[80] Xue Jiang, Jianhui Zhao, Bo Du, and Zhiyong Yuan. Self-supervised contrastive learning for eeg-based sleep staging. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[81] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058, 2020.

[82] Frans F Jobsis. Noninvasive, infrared monitoring of cerebral and myocardial oxygen sufficiency and circulatory parameters. *Science*, 198(4323):1264–1267, 1977.

[83] CS Jordan and SD Brennen. Instantaneous self-assessment of workload technique (isa). *Defence Research Agency, Portsmouth*, 1992.

[84] Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in ml-based science. *arXiv preprint arXiv:2207.07048*, 2022.

[85] Ivan Kesedžić, Marko Šarlija, Jelena Božek, Siniša Popović, and Krešimir Ćosić. Classification of cognitive load based on neurophysiological features from functional near-infrared spectroscopy and electrocardiography signals on n-back task. *IEEE Sensors Journal*, 2020.

[86] Khurram Khalil, Umer Asgher, and Yasar Ayaz. Novel fnirs study on homogeneous symmetric feature-based transfer learning for brain–computer interface. *Scientific reports*, 12(1):3198, 2022.

[87] MN Afzal Khan and Keum-Shik Hong. Most favorable stimulation duration in the sensorimotor cortex for fnirs-based bci. *Biomedical Optics Express*, 12(10):5939–5954, 2021.

[88] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[89] Evgeniya Kirilina, Na Yu, Alexander Jelzow, Heidrun Wabnitz, Arthur M Jacobs, and Ilias Tachtsidis. Identifying and quantifying main components of physiological noise in functional near infrared spectroscopy on the prefrontal cortex. *Frontiers in human neuroscience*, 7:864, 2013.

[90] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[91] Jasmine Kwasa, Hannah M Peterson, Kavon Karrobi, Lietsel Jones, Termara Parker, Nia Nickerson, and Sossena Wood. Demographic reporting and phenotypic exclusion in fnirs. *Frontiers in Neuroscience*, 17:1086208, 2023.

[92] Tian Lan, Deniz Erdogmus, Andre Adami, Santosh Mathan, and Misha Pavel. Channel selection and feature projection for cognitive load estimation using ambulatory eeg. *Computational intelligence and neuroscience*, 2007, 2007.

[93] Aleksandra Landowska, David Roberts, Peter Eachus, and Alan Barrett. Within-and between-session prefrontal cortex response to virtual reality exposure therapy for acrophobia. *Frontiers in human neuroscience*, 12:362, 2018.

[94] Archana Laxmisan, Forogh Hakimzada, Osman R Sayan, Robert A Green, Jiajie Zhang, and Vimla L Patel. The multitasking clinician: decision-making and cognitive demand during and after team handoffs in emergency care. *International journal of medical informatics*, 76(11-12):801–811, 2007.

[95] Anh Son Le, Hirofumi Aoki, Fumihiko Murase, and Kenji Ishida. A novel method for classifying driver mental workload under naturalistic conditions with information from near-infrared spectroscopy. *Frontiers in human neuroscience*, 12:431, 2018.

[96] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[97] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[98] Lam Ghai Lim, Wei Chun Ung, Yee Ling Chan, Cheng Kai Lu, Stephanie Sutoko, Tsukasa Funane, Masashi Kiguchi, and Tong Boon Tang. A unified analytical framework with multiple fnirs features for mental workload assessment in the prefrontal cortex. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2020.

[99] Martin A Lindquist, Ji Meng Loh, Lauren Y Atlas, and Tor D Wager. Modeling the hemodynamic response function in fmri: efficiency, bias and mis-modeling. *Neuroimage*, 45(1):S187–S198, 2009.

[100] Ruixue Liu, Bryan Reimer, Siyang Song, Bruce Mehler, and Erin Solovey. Unsupervised fnirs feature extraction with cae and esn autoencoder for driver cognitive load classification. *Journal of Neural Engineering*, 18(3):036002, 2021.

[101] Ruixue Liu, Erin Walker, Leah Friedman, Catherine M Arrington, and Erin T Solovey. fnirs-based classification of mind-wandering with personalized window selection for multimodal learning interfaces. *Journal on multimodal user interfaces*, 15:257–272, 2021.

[102] Ignacio Lucas, Patrícia Urieta, Ferran Balada, Eduardo Blanco, and Anton Aluja. Differences in prefrontal cortex activity based on difficulty in a working memory task using near-infrared spectroscopy. *Behavioural Brain Research*, page 112722, 2020.

[103] Robert Luke, Eric D Larson, Maureen J Shader, Hamish Innes-Brown, Lindsey Van Yper, Adrian KC Lee, Paul F Sowman, and David McAlpine. Analysis methods for measuring passive auditory fnirs responses generated by a block-design paradigm. *Neurophotonics*, 8(2):025008, 2021.

[104] Boyang Lyu, Thao Pham, Giles Blaney, Zachary Haga, Sergio Fantini, and Shuchin Aeron. Domain adaptation for robust workload classification using fnirs. *arXiv preprint arXiv:2007.06706*, 2020.

[105] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[106] Horia A Maior, Matthew Pike, Sarah Sharples, and Max L Wilson. Examining the reliability of using fnirs in realistic hci settings for spatial and verbal tasks. In *proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 3039–3042, 2015.

[107] Horia A Maior, Matthew Pike, Max L Wilson, and Sarah Sharples. Continuous detection of workload overload: An fnirs approach. In *Contemporary Ergonomics and Human Factors 2014: Proceedings of the international conference on Ergonomics & Human Factors 2014, Southampton, UK, 7-10 April 2014*, page 450. CRC Press, 2014.

[108] Horia A Maior, Max L Wilson, and Sarah Sharples. Workload alerts—using physiological measures of mental workload to provide feedback during tasks. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 25(2):9, 2018.

[109] Adrian Cornelius Marinescu, Sarah Sharples, Alastair Campbell Ritchie, Tomas Sanchez Lopez, Michael McDowell, and Hervé P Morvan. Physiological parameter response to variation of mental workload. *Human factors*, 60(1):31–56, 2018.

[110] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[111] Ryan McKendrick, Bradley Feest, Amanda Harwood, and Brian Falcone. Theories and methods for labeling cognitive workload: Classification and transfer learning. *Frontiers in human neuroscience*, 13:295, 2019.

[112] Kai J Miller, Gerwin Schalk, Eberhard E Fetz, Marcel den Nijs, Jeffrey G Ojemann, and Rajesh PN Rao. Cortical activity during motor execution, motor imagery, and imagery-based online feedback. *Proceedings of the National Academy of Sciences*, 107(9):4430–4435, 2010.

[113] Tom M Mitchell. Machine learning, 1997.

[114] Erika Molteni, Michele Butti, Anna M Bianchi, and Gianluigi Reni. Activation of the prefrontal cortex during a visual n-back working memory task with varying memory load: a near infrared spectroscopy study. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4024–4027. IEEE, 2008.

[115] John Mongan, Linda Moy, and Charles E Kahn Jr. Checklist for artificial intelligence in medical imaging (claim): a guide for authors and reviewers. *Radiology. Artificial Intelligence*, 2(2), 2020.

[116] Tomoyuki Nagasawa, Takanori Sato, Isao Nambu, and Yasuhiro Wada. Improving fnirs-bci accuracy using gan-based data augmentation. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1208–1211. IEEE, 2019.

[117] Tomoyuki Nagasawa, Takanori Sato, Isao Nambu, and Yasuhiro Wada. fnirs-gans: Data augmentation using generative adversarial networks for classifying motor tasks from functional near-infrared spectroscopy. *Journal of Neural Engineering*, 2020.

[118] Masaki Nakanishi, Minpeng Xu, Yijun Wang, Kuan-Jung Chiang, Jin Han, and Tzyy-Ping Jung. Questionable classification accuracy reported in "designing a sum of squared correlations framework for enhancing ssvep-based bcis". *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(4):1042–1043, 2020.

[119] Noman Naseer and Keum-Shik Hong. fnirs-based brain-computer interfaces: a review. *Frontiers in human neuroscience*, 9:3, 2015.

[120] Hammad Nazeer, Noman Naseer, Rayyan Azam Azam Khan, Farzan Majeed Noori, Nauman Khalid Qureshi, Umar Shahbaz Khan, and Muhammad Jawad Khan. Enhancing classification accuracy of fnirs-bci using features acquired from vector-based phase analysis. *Journal of Neural Engineering*, 2020.

[121] Peter Nickel and Friedhelm Nachreiner. Sensitivity and diagnosticity of the 0.1-hz component of heart rate variability as an indicator of mental workload. *Human factors*, 45(4):575–590, 2003.

[122] Ryota Nishiyori. fnirs: An emergent method to document functional cortical activity during infant movements. *Frontiers in psychology*, 7:533, 2016.

[123] Takayuki Nozawa and Yoshihiro Miyake. Capturing individual differences in prefrontal activity with wearable fnirs for daily use. In *2020 13th International Conference on Human System Interaction (HSI)*, pages 249–254. IEEE, 2020.

[124] Calvin KL Or and Vincent G Duffy. Development of a facial skin temperature-based methodology for non-intrusive mental workload measurement. *Occupational Ergonomics*, 7(2):83–94, 2007.

[125] Adrian M Owen, Kathryn M McMillan, Angela R Laird, and Ed Bullmore. N-back working memory paradigm: A meta-analysis of normative functional neuroimaging studies. *Human brain mapping*, 25(1):46–59, 2005.

[126] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[127] Evan M M Peck, Beste F Yuksel, Alvitta Ottley, Robert JK Jacob, and Remco Chang. Using fnirs brain sensing to evaluate information visualization interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 473–482, 2013.

[128] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[129] Gert Pfurtscheller and Christa Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001.

[130] David Picard. Torch. manual_seed (3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision. *arXiv preprint arXiv:2109.08203*, 2021.

[131] Matthew F Pike, Horia A Maior, Martin Porcheron, Sarah C Sharples, and Max L Wilson. Measuring the effect of think aloud protocols on workload using fnirs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3807–3816, 2014.

[132] Paola Pinti, Clarisse Aichelburg, Sam Gilbert, Antonia Hamilton, Joy Hirsch, Paul Burgess, and Ilias Tachtsidis. A review on the use of wearable functional near-infrared spectroscopy in naturalistic environments. *Japanese Psychological Research*, 60(4):347–373, 2018.

[133] Paola Pinti, Daniela Cardone, and Arcangelo Merla. Simultaneous fnirs and thermal infrared imaging during cognitive task reveal autonomic correlates of prefrontal cortex activity. *Scientific reports*, 5(1):1–14, 2015.

[134] Paola Pinti, Felix Scholkmann, Antonia Hamilton, Paul Burgess, and Ilias Tachtsidis. Current status and issues regarding pre-processing of fnirs neuroimaging data: An investigation of diverse signal filtering methods within a general linear model framework. *Frontiers in human neuroscience*, 12:505, 2018.

[135] Paola Pinti, MF Siddiqui, AD Levy, EJH Jones, and Ilias Tachtsidis. An analysis framework for the integration of broadband nirs and eeg to assess neurovascular and neurometabolic coupling. *Scientific reports*, 11(1):3977, 2021.

[136] Paola Pinti, Ilias Tachtsidis, Antonia Hamilton, Joy Hirsch, Clarisse Aichelburg, Sam Gilbert, and Paul W Burgess. The present and future use of functional near-infrared spectroscopy (fnirs) for cognitive neuroscience. *Annals of the New York Academy of Sciences*, 2018.

[137] Nicolas Pinto, David Doukhan, James J DiCarlo, and David D Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS computational biology*, 5(11):e1000579, 2009.

[138] Russell A Poldrack. Region of interest analysis for fmri. *Social cognitive and affective neuroscience*, 2(1):67–70, 2007.

[139] Scott Prahl. Tabulated molar extinction coefficient for hemoglobin in water. `https://omlc.org/spectra/hemoglobin/index.html`, 1998. Accessed: 2019-05-20.

[140] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[141] Md Asadur Rahman, Mohd Abdur Rashid, and Mohiuddin Ahmad. Selecting the optimal conditions of savitzky–golay filter for fnirs signal. *Biocybernetics and Biomedical Engineering*, 39(3):624–637, 2019.

[142] James Reason. The contribution of latent human failures to the breakdown of complex systems. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 327(1241):475–484, 1990.

[143] Raphaëlle N Roy, Marcel F Hinss, Ludovic Darmet, Simon Ladouce, Emilie S Jahanpour, Bertille Somon, Xiaoqi Xu, Nicolas Drougard, Frédéric Dehais, and Fabien Lotte. Retrospective on the first passive brain-computer interface competition on cross-session workload estimation. *Frontiers in Neuroergonomics*, 3:838342, 2022.

[144] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[145] Mohammed Rupawala, Hamid Dehghani, Samuel JE Lucas, Peter Tino, and Damian Cruse. Shining a light on awareness: A review of functional near-infrared spectroscopy for prolonged disorders of consciousness. *Frontiers in neurology*, 9:350, 2018.

[146] Marjan Saadati, Jill Nelson, and Hasan Ayaz. Convolutional neural network for hybrid fnirs-eeg mental workload classification. In *International Conference on Applied Human Factors and Ergonomics*, pages 221–232. Springer, 2019.

[147] Hendrik Santosa, Xuetong Zhai, Frank Fishburn, Patrick J Sparto, and Theodore J Huppert. Quantitative comparison of correction techniques for removing systemic physiological signal in functional near-infrared spectroscopy studies. *Neurophotonics*, 7(3):035009, 2020.

[148] Takanori Sato, Isao Nambu, Kotaro Takeda, Takatsugu Aihara, Okito Yamashita, Yuko Isogaya, Yoshihiro Inoue, Yohei Otaka, Yasuhiro Wada, Mitsuo Kawato, et al. Reduction of global interference of scalp-hemodynamics in functional near-infrared spectroscopy using short distance probes. *NeuroImage*, 141:120–132, 2016.

[149] Robin M Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley–benchmarking deep learning optimizers. *arXiv preprint arXiv:2007.01547*, 2020.

[150] Felix Scholkmann, Stefan Kleiser, Andreas Jaakko Metz, Raphael Zimmermann, Juan Mata Pavia, Ursula Wolf, and Martin Wolf. A review on continuous wave functional near-infrared spectroscopy and imaging instrumentation and methodology. *Neuroimage*, 85:6–27, 2014.

[151] Matthias L Schroeter, Thomas Kupka, Toralf Mildner, Kâmil Uludağ, and D Yves von Cramon. Investigating the post-stimulus undershoot of the bold signal—a simultaneous fmri and fnirs study. *Neuroimage*, 30(2):349–358, 2006.

[152] Yu Shi, Natalie Ruiz, Ronnie Taib, Eric Choi, and Fang Chen. Galvanic skin response (gsr) as an index of cognitive load. In *CHI'07 extended abstracts on Human factors in computing systems*, pages 2651–2656, 2007.

[153] Jaeyoung Shin, Klaus-R Müller, and Han-Jeong Hwang. Near-infrared spectroscopy (nirs)-based eyes-closed brain-computer interface (bci) using prefrontal cortex activation due to mental arithmetic. *Scientific reports*, 6:36203, 2016.

[154] Jaeyoung Shin, Alexander von Lühmann, Benjamin Blankertz, Do-Won Kim, Jichai Jeong, Han-Jeong Hwang, and Klaus-Robert Müller. Open access dataset for eeg+nirs single-trial classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(10):1735–1745, 2016.

[155] Jaeyoung Shin, Alexander Von Lühmann, Do-Won Kim, Jan Mehnert, Han-Jeong Hwang, and Klaus-Robert M"uller. Simultaneous acquisition of eeg and nirs during cognitive tasks for an open access dataset. *Scientific data*, 5:180003, 2018.

[156] Ranganatha Sitaram, Haihong Zhang, Cuntai Guan, Manoj Thulasidas, Yoko Hoshi, Akihiro Ishikawa, Koji Shimizu, and Niels Birbaumer. Temporal classification of multichannel near-infrared spectroscopy signals of motor imagery for developing a brain–computer interface. *NeuroImage*, 34(4):1416–1427, 2007.

[157] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[158] Erin Treacy Solovey, Audrey Girouard, Krysta Chauncey, Leanne M Hirshfield, Angelo Sassaroli, Feng Zheng, Sergio Fantini, and Robert JK Jacob. Using fnirs brain sensing in realistic hci settings: experiments and guidelines. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 157–166, 2009.

[159] Jens Steinbrink, Arno Villringer, Florian Kempf, Daniel Haux, Stefanie Boden, and Hellmuth Obrig. Illuminating the bold signal: combined fmri–fnirs studies. *Magnetic resonance imaging*, 24(4):495–505, 2006.

[160] Gary Strangman, Joseph P Culver, John H Thompson, and David A Boas. A quantitative comparison of simultaneous bold fmri and nirs recordings during functional brain activation. *Neuroimage*, 17(2):719–731, 2002.

[161] Andrew J Tattersall and Penelope S Foord. An experimental evaluation of instantaneous self-assessment as a measure of workload. *Ergonomics*, 39(5):740–748, 1996.

[162] Thanawin Trakoolwilaiwan, Bahareh Behboodi, Jaeseok Lee, Kyungsoo Kim, and Ji-Woong Choi. Convolutional neural network for high-accuracy functional near-infrared spectroscopy in a brain–computer interface: three-class classification of rest, right-, and left-hand motor execution. *Neurophotonics*, 5(1):011008, 2017.

[163] Anirudh Unni, Klas Ihme, Meike Jipp, and Jochem W Rieger. Assessing the driver's current level of working memory load with high density functional near-infrared spectroscopy: a realistic driving simulator study. *Frontiers in human neuroscience*, 11:167, 2017.

[164] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[165] Anouk Vermeij, Roy PC Kessels, Linda Heskamp, Esther MF Simons, Paul LJ Dautzenberg, and Jurgen AHR Claassen. Prefrontal activation may predict working-memory training gain in normal aging and mild cognitive impairment. *Brain Imaging and Behavior*, 11:141–154, 2017.

[166] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[167] Rodrigo Vitorio, Sam Stuart, Lynn Rochester, Lisa Alcock, and Annette Pantall. fnirs response during walking—artefact or cortical activity? a systematic review. *Neuroscience & Biobehavioral Reviews*, 83:160–172, 2017.

[168] Jiyang Wang, Trevor Grant, Senem Velipasalar, Baocheng Geng, and Leanne Hirshfield. Taking a deeper look at the brain: predicting visual perceptual and working memory load from high-density fnirs data. *IEEE Journal of Biomedical and Health Informatics*, 26(5):2308–2319, 2021.

[169] Shouyi Wang, Jacek Gwizdka, and W Art Chaovalitwongse. Using wireless eeg signals to assess memory workload in the n-back task. *IEEE Transactions on Human-Machine Systems*, 46(3):424–435, 2015.

[170] Robert Whelan and Hugh Garavan. When optimism hurts: inflated predictions in psychiatric neuroimaging. *Biological psychiatry*, 75(9):746–748, 2014.

[171] Bertram Wortelen, Anirudh Unni, Jochem W Rieger, and Andreas L"udtke. Towards the integration and evaluation of online workload measures in a cognitive architecture. In *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 000011–000016. IEEE, 2016.

[172] Susan Wray, Mark Cope, David T Delpy, John S Wyatt, and E Osmund R Reynolds. Characterization of the near infrared absorption spectra of cytochrome aa3 and haemoglobin for the non-invasive monitoring of cerebral oxygenation. *Biochimica et Biophysica Acta (BBA)-Bioenergetics*, 933(1):184–192, 1988.

[173] Dongrui Wu, Yifan Xu, and Bao-Liang Lu. Transfer learning for eeg-based brain–computer interfaces: A review of progress made since 2016. *IEEE Transactions on Cognitive and Developmental Systems*, 14(1):4–19, 2020.

[174] So-Hyeon Yoo, Seong-Woo Woo, and Zafar Amad. Classification of three categories from prefrontal cortex using lstm networks: fnirs study. In *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 1141–1146. IEEE, 2018.

[175] Mark S Young, Karel A Brookhuis, Christopher D Wickens, and Peter A Hancock. State of science: mental workload in ergonomics. *Ergonomics*, 58(1):1–17, 2015.

[176] Meryem A Yücel, Alexander v Lühmann, Felix Scholkmann, Judit Gervain, Ippeita Dan, Hasan Ayaz, David Boas, Robert J Cooper, Joseph Culver, Clare E Elwell, et al. Best practices for fnirs publications. *Neurophotonics*, 8(1):012101, 2021.

[177] Meryem A Yücel, Juliette Selb, David A Boas, Sydney S Cash, and Robert J Cooper. Reducing motion artifacts for long-term clinical nirs monitoring using collodion-fixed prism-based optical fibers. *Neuroimage*, 85:192–201, 2014.

[178] Beste F Yuksel, Kurt B Oleson, Lane Harrison, Evan M Peck, Daniel Afergan, Remco Chang, and Robert JK Jacob. Learn piano with bach: An adaptive learning interface that adjusts task difficulty based on brain state. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 5372–5384, 2016.

[179] Amad Zafar and Keum-Shik Hong. Detection and classification of three-class initial dips from prefrontal cortex. *Biomedical optics express*, 8(1):367–383, 2017.

[180] Thorsten O Zander and Christian Kothe. Towards passive brain–computer interfaces: applying brain–computer interface technology to human–machine systems in general. *Journal of neural engineering*, 8(2):025005, 2011.

[181] Phillips V Zephaniah and Jae Gwan Kim. Recent functional near infrared spectroscopy based brain computer interface systems: developments, applications and challenges. *Biomedical Engineering Letters*, 4(3):223–230, 2014.

[182] Fred RH Zijlstra, Robert A Roe, Anna B Leonora, and Irene Krediet. Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology*, 72(2):163–185, 1999.

[183] WG Zijlstra, A Buursma, and WP Meeuwsen-Van der Roest. Absorption spectra of human fetal and adult oxyhemoglobin, de-oxyhemoglobin, carboxyhemoglobin, and methemoglobin. *Clinical chemistry*, 37(9):1633–1638, 1991.

Appendices

A *NIRSimple* documentation (Chapter 3)

preprocessing module

The preprocessing module contains functions to preprocess fNIRS signals.

preprocessing.intensities_to_od_changes(*intensities*, *refs=None*)

Converts intensities into optical density changes. Changes are relative to the average intensity or a reference intensity for each channel.

Optical density from light intensity: OD = log10(I_0/I_t)

Optical density changes relative to average transmitted intensity: delta_OD = log10(I_0/I_t) - log10(I_0/I_average) delta_OD = -log10(I_t/I_average)

Parameters:
- **intensities** (*array*) – numpy array of absolute intensities, must have the correct shape (channels, data points).
- **refs** (*list of floats*) – List of reference intensities to use instead of averages, length must be equal to the number of channels.

Returns: **delta_od** – numpy array of optical density changes, relative to average intensities or a reference for each channel, of shape (channels, data points).

Return type: array

preprocessing.mbll(*delta_od*, *ch_names*, *ch_wls*, *ch_dpfs*, *ch_distances*, *unit*, *table='wray'*)

Apply the modified Beer-Lambert law (from Delpy et al., 1988) to optical density changes in order to obtain concentration changes in oxygenated hemoglobin (HbO) and deoxygenated hemoglobin (HbR).

Modified Beer-Lambert law: Dod_wl = e_HbO_wl*Dc_HbO*l*DPF_wl + e_HbR_wl*Dc_HbR*l*DPF_wl

With two different wavelengths we obtain a set of linear equations solved with matrices: [[Dod_1] = [[e_HbO_1*l*DPF_1, e_HbR_1*l*DPF_1] . [[Dc_HbO] [Dod_2]] [e_HbO_2*l*DPF_2, e_HbR_2*l*DPF_2]] [Dc_HbR]]

Equivalent to: [[Dc_HbO] = [[e_HbO_1, e_HbR_1] -1 . [[Dod_1/(l*DPF_1)] [DC_HbR]] [e_HbO_2, e_HbR_2]] [Dod_2/(l*DPF_2)]]

Parameters:

- **delta_od** (*array*) – numpy array of optical density changes, relative to average intensities for each channel, of shape (channels, data points).
- **ch_names** (*list of strings*) – List of channel names.
- **ch_wls** (*list of integers*) – List of channel wavelengths (in nm).
- **ch_dpfs** (*list of floats*) – List of channel differential pathlength factors (DPF) (also called partial pathlengths factors (PPF)).
- **ch_distances** (*list of floats*) – List of channel source-detector distances.
- **unit** (*string*) – Unit for ch_distances ('cm' or 'mm').
- **table** (*string*) – Table to use as molar extinction coefficients. 'wray': data from S. Wray et al., 1988 'cope': data from M. Cope, 1991 'gratzer': data from W.B. Gratzer and K. Kollias compiled by S. Prahl 'moaveni': data from M.K. Moaveni and J.M. Schmitt compiled by S. Prahl 'takatani': data from S. Takatani and M.D. Graham compiled by S. Prahl

Returns:

- **delta_c** (*array*) – numpy array of hemoglobin concentration changes in [moles/liter] or [M] for each channel, of shape (channels, data points).
- **new_ch_names** (*list of strings*) – New list of channel names.
- **new_ch_types** (*list of strings*) – New list of channel types ('hbo' for oxygenated hemoglobin and 'hbr' for deoxygenated hemoglobin).

`preprocessing.od_to_od_changes(optical_densities, refs=None)`

Convert optical densities into optical density changes, relative to the average optical density or a reference optical density for each channel.

Optical density changes relative to average optical density: delta_OD = OD - OD_average

Parameters:

- **optical_densities** (*array*) – numpy array of optical densities, must have the correct shape (channels, data points).
- **refs** (*list of floats*) – List of reference optical densities to use instead of averages, length must be equal to the number of channels.

Returns: **delta_od** – numpy array of optical density changes, relative to average optical densities or a reference for each channel, of shape (channels, data points).

Return type: array

processing module

The processing module contains functions to process fNIRS signals.

processing.cbsi(*delta_c, ch_names, ch_types*)

Apply correlation based signal improvement (from Cui at al., 2010) to hemoglobin concentration changes.

Correlation based signal improvement (CBSI): x_0 = (1/2)*(x-alpha*x) y_0 = -(1/alpha)*x_0

Parameters:

- **delta_c** (*array*) – numpy array of hemoglobin concentration changes in [moles/liter] or [M] for each channel, of shape (channels, data points).
- **ch_names** (*list of strings*) – List of channel names.
- **ch_types** (*list of strings*) – List of channel types ('hbo' for oxygenated hemoglobin and 'hbr' for deoxygenated hemoglobin).

Returns:

- **delta_c_0** (*array*) – numpy array of corrected activation signals in [moles/liter] or [M] for each channel, of shape (channels, data points).
- **new_ch_names** (*list of strings*) – New list of channel names.
- **new_ch_types** (*list of strings*) – New list of channel types ('hbo' for oxygenated hemoglobin and 'hbr' for deoxygenated hemoglobin).

B *BenchNIRS* documentation (Chapter 3 and 6)

load module

load.load_dataset(*dataset*, *path=None*, *bandpass=None*, *order=4*, *tddr=False*, *baseline=(None, 0)*, *roi_sides=False*)

Load and filter one of the open access dataset.

Parameters:

- **dataset** (*string*) – Dataset to load. `'herff_2014_nb'` for n-back from Herff et al., 2014 (epoch interval: -5 to 44 seconds). `'shin_2018_nb'` for n-back from Shin et al., 2018 (epoch interval: -2 to 40 seconds). `'shin_2018_wg'` for word generation from Shin et al., 2018 (epoch interval: -2 to 10 seconds). `'shin_2016_ma'` for mental arithmetic from Shin et al., 2016 (epoch interval: -2 to 10 seconds). `'bak_2019_me'` for motor execution from Bak et al., 2019 (epoch interval: -2 to 10 seconds).
- **path** (*string* | *None*) – Path of the dataset selected with the `dataset` parameter. Defaults to `None` to use the default path.
- **bandpass** (*list of floats* | *None*) – Cutoff frequencies of the bandpass Butterworth filter. Defaults to `None` for no filtering.
- **order** (*integer*) – Order of the bandpass Butterworth filter.
- **tddr** (*boolean*) – Whether to apply temporal derivative distribution repair.
- **baseline** (*None or tuple of length 2*) – The time interval to apply baseline correction. If `None` do not apply it. If a tuple `(a, b)` the interval is between `a` and `b` (in seconds). If `a` is `None` the beginning of the data is used and if `b` is `None` then `b` is set to the end of the interval. If `(None, None)` all the time interval is used. Correction is applied by computing mean of the baseline period and subtracting it from the data. The baseline `(a, b)` includes both endpoints, i.e. all timepoints `t` such that `a <= t <= b`.
- **roi_sides** (*boolean*) – Whether to average channels by side.

Returns: **epochs** – MNE epochs of filtered data with associated labels, downsampled to 10 Hz for comparison purposes. Subject IDs are contained in the `metadata` property.

Return type: MNE Epochs object

process module

process.process_epochs(*mne_epochs*, *tmax=None*, *tslide=None*, *sort=False*, *reject_criteria=None*)

Perform processing on epochs including baseline cropping, bad epoch removal, label extraction and unit conversion.

Parameters:

- **mne_epochs** (*MNE Epochs object*) – MNE epochs of filtered data with associated labels. Subject IDs are contained in the `metadata` property.
- **tmax** (*float* | *None*) – End time of selection in seconds. Defaults to `None` to keep the initial end time.
- **tslide** (*float* | *None*) – Size of the sliding window in seconds. Will crop the epochs if `tmax` is not a multiple of `tslide`. Defaults to `None` for no window sliding.
- **sort** (*boolean*) – Whether to sort channels by type (all HbO, all HbR). Defaults to `False` for no sorting.
- **reject_criteria** (*list of floats* | *None*) – List of the 2 peak-to-peak rejection thresholds for HbO and HbR channels respectively in uM. Defaults to `None` for no rejection.

Returns:

- **nirs** (*array of shape (n_epochs, n_channels, n_times)*) – Processed NIRS data in uM.
- **labels** (*array of integer*) – List of labels.
- **groups** (*array of integer*) – List of subject ID matching the epochs.

learn module

learn.deep_learn(*nirs, labels, groups, model_class, features=None, normalize=False, batch_sizes=[4, 8, 16, 32, 64], lrs=[1e-05, 0.0001, 0.001, 0.01, 0.1], max_epoch=100, random_state=None, output_folder='./outputs'*)

Perform nested k-fold cross-validation for a deep learning model. Produces loss graph, accuracy graph and confusion matrice. This is made with early stopping (with a validation set of 20 %) once the model has been fine tuned on the inner loop of the nested k-fold cross-validation. The number of classes is deduced from the number of unique labels.

Parameters:

- **nirs** (*array of shape (n_epochs, n_channels, n_times)*) – Processed NIRS data.
- **labels** (*array of integers*) – List of labels matching the NIRS data.
- **groups** (*array of integers | None*) – List of subject ID matching the epochs to perfrom a group k-fold cross-validation. If `None`, performs a stratified k-fold cross-validation instead.
- **model_class** (*string | PyTorch nn.Module class*) – The PyTorch model class to use. If a string, can be either `'ann'`, `'cnn'` or `'lstm'`. If a PyTorch `nn.Module` class, the `__init__()` method must accept the number of classes as a parameter, and this needs to be the number of output neurons.
- **features** (*list of strings | None*) – List of features to extract. The list can include `'mean'` for the mean along the time axis, `'std'` for standard deviation along the time axis and `'slope'` for the slope of the linear regression along the time axis. Defaults to `None` for no feature extration and using the raw data.
- **normalize** (*boolean*) – Whether to normalize data before feeding to the model with min-max scaling based on the train set for each iteration of the outer cross-validation. Defaults to `False` for no normalization.
- **batch_sizes** (*list of integers*) – List of batch sizes to test for optimization.
- **lrs** (*list of floats*) – List of learning rates to test for optimization.
- **max_epoch** (*integer*) – Maximum number of epochs possible for training. Defaults to `100`.
- **random_state** (*integer | None*) – Controls the shuffling applied to data and random model initialization. Pass an integer for reproducible output across multiple function calls. Defaults to `None` for not setting the seed.
- **output_folder** (*string*) – Path to the directory into which the figures will be saved. Defaults to `'./outputs'`.

Returns:

- **accuracies** (*list of floats*) – List of accuracies on the test sets (one for each iteration of the outer cross-validation).
- **all_hps** (*list of tuples*) – List of hyperparameters (one tuple for each iteration of the outer cross-validation). Each tuple will be *(batch size, learning rate)*.
- **additional_metrics** (*list of tuples*) – List of tuples of metrics composed of (precision, recall, F1 score) on the outer cross-validation (one tuple for each iteration of the outer cross-validation). This uses the `precision_recall_fscore_support` function from scikit-learn with `average='weighted'`, `y_true` and `y_pred` being the true and the predictions on the specific iteration of the outer cross-validation.

`learn.deep_transfer_learn`(*nirs, labels, groups, enc_class, dec_class, model_class, features=None, normalize=False, batch_sizes=[4, 8, 16, 32, 64], lrs=[1e-05, 0.0001, 0.001, 0.01, 0.1], max_epoch=100, random_state=None, output_folder='./outputs'*)

Perform nested k-fold cross-validation for a deep transfer learning model. Produces loss graphs, accuracy graph and confusion matrice. This is made with early stopping (with a validation set of 20 %) once the model has been fine tuned on the inner loop of the nested k-fold cross-validation. The number of classes is deduced from the number of unique labels.

Parameters:

- **nirs** (*array of shape (n_epochs, n_channels, n_times)*) – Processed NIRS data.
- **labels** (*array of integers*) – List of labels matching the NIRS data. Must include values 8 for all the unlabeled segments this should be performed by *process_epochs*.
- **groups** (*array of integers | None*) – List of subject ID matching the epochs to perfrom a group k-fold cross-validation. If `None`, performs a stratified k-fold cross-validation instead.
- **enc_class** (*PyTorch nn.Module class*) – The PyTorch encoder class to use for each Hb channel type.
- **dec_class** (*PyTorch nn.Module class*) – The PyTorch decoder class to use for each Hb channel type.
- **model_class** (*PyTorch nn.Module class*) – The PyTorch classifier class to use for the overall classification containing the Hb encoders. The `__init__()` method must accept the number of classes, an encoder for HbO and an encoder for HbR as parameters. The number of classes needs to be the number of output neurons.
- **features** (*list of strings | None*) – List of features to extract. The list can include `'mean'` for the mean along the time axis, `'std'` for standard deviation along the time axis and `'slope'` for the slope of the linear regression along the time axis. Defaults to `None` for no feature extration and using the raw data.
- **normalize** (*boolean*) – Whether to normalize data before feeding to the model with min-max scaling based on the train set for each iteration of the outer cross-validation. Defaults to `False` for no normalization.
- **batch_sizes** (*list of integers*) – List of batch sizes to test for optimization.
- **lrs** (*list of floats*) – List of learning rates to test for optimization.
- **max_epoch** (*integer*) – Maximum number of epochs possible for training. Defaults to `100`.
- **random_state** (*integer | None*) – Controls the shuffling applied to data and random model initialization. Pass an integer for reproducible output across multiple function calls. Defaults to `None` for not setting the seed.
- **output_folder** (*string*) – Path to the directory into which the figures will be saved. Defaults to `'./outputs'`.

Returns:

- **accuracies** (*list of floats*) – List of accuracies for the overall classifier on the test sets (one for each iteration of the outer cross-validation).
- **all_hps** (*list of tuples*) – List of hyperparameters for the overall classifier (one tuple for each iteration of the outer cross-validation). Each tuple will be *(batch size, learning rate)*.
- **additional_metrics** (*list of tuples*) – List of tuples of metrics composed of (precision, recall, F1 score) on the outer cross-validation (one tuple for each iteration of the outer cross-validation). This uses the `precision_recall_fscore_support` function from scikit-learn with `average='weighted'`, `y_true` and `y_pred` being the true and the predictions on the specific iteration of the outer cross-validation.

learn.machine_learn(*nirs, labels, groups, model, features, normalize=False, random_state=None, output_folder='./outputs'*)

Perform nested k-fold cross-validation for standard machine learning models producing metrics and confusion matrices. The models include linear discriminant analysis (LDA), support vector classifier (SVC) with grid search for the regularization parameter (inner cross-validation), and k-nearest neighbors (kNN) with grid search for the number of neighbors (inner cross-validation).

Parameters:

- **nirs** (*array of shape (n_epochs, n_channels, n_times)*) – Processed NIRS data.
- **labels** (*array of integers*) – List of labels matching the NIRS data.
- **groups** (*array of integers | None*) – List of subject ID matching the epochs to perfrom a group k-fold cross-validation. If `None`, performs a stratified k-fold cross-validation instead.
- **model** (*string*) – Standard machine learning to use. Either `'lda'` for a linear discriminant analysis, `'svc'` for a linear support vector classifier or `'knn'` for a k-nearest neighbors classifier.
- **features** (*list of strings*) – List of features to extract. The list can include `'mean'` for the mean along the time axis, `'std'` for standard deviation along the time axis and `'slope'` for the slope of the linear regression along the time axis.
- **normalize** (*boolean*) – Whether to normalize data before feeding to the model with min-max scaling based on the train set for each iteration of the outer cross-validation. Defaults to `False` for no normalization.
- **random_state** (*integer | None*) – Controls the shuffling applied to data. Pass an integer for reproducible output across multiple function calls. Defaults to `None` for not setting the seed.
- **output_folder** (*string*) – Path to the directory into which the figures will be saved. Defaults to `'./outputs'`.

Returns:

- **accuracies** (*list of floats*) – List of accuracies on the test sets (one for each iteration of the outer cross-validation).
- **all_hps** (*list of floats | list of None*) – List of regularization parameters for the SVC or a list of None for the LDA (one for each iteration of the outer cross-validation).
- **additional_metrics** (*list of tuples*) – List of tuples of metrics composed of (precision, recall, F1 score) on the outer cross-validation (one tuple for each iteration of the outer cross-validation). This uses the `precision_recall_fscore_support` function from scikit-learn with `average='weighted'`, `y_true` and `y_pred` being the true and the predictions on the specific iteration of the outer cross-validation.

viz module

viz.epochs_viz(*mne_epochs*, *reject_criteria=None*)

Perform processing and visualization on epochs. Processing includes baseline cropping and bad epoch removal.

Parameters:

- **mne_epochs** (*MNE Epochs object*) – MNE epochs of filtered data with associated labels. Subject IDs are contained in the `metadata` property.
- **reject_criteria** (*list of floats | None*) – List of the 2 peak-to-peak rejection thresholds for HbO and HbR channels respectively in uM. Defaults to `None` for no rejection.

C Confusion matrices (Chapter 4)

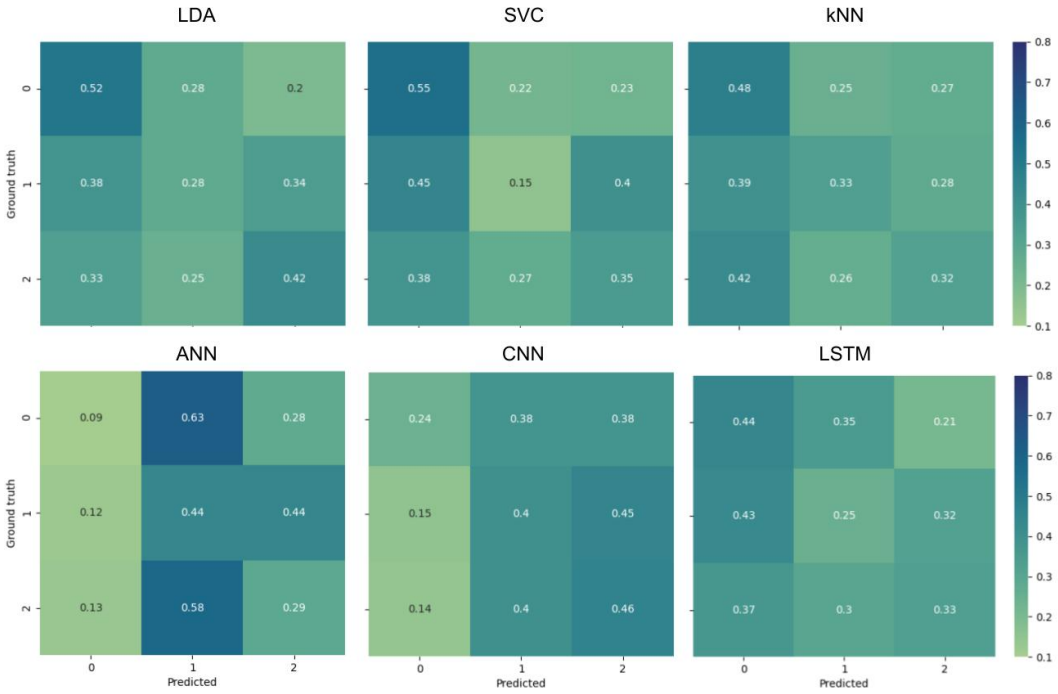

Figure 1: Confusion matrices for the subject-independent approach on the Herff et al. 2014 dataset of n-back tasks. Classes 0, 1 and 2 are 1-, 2- and 3-back respectively.

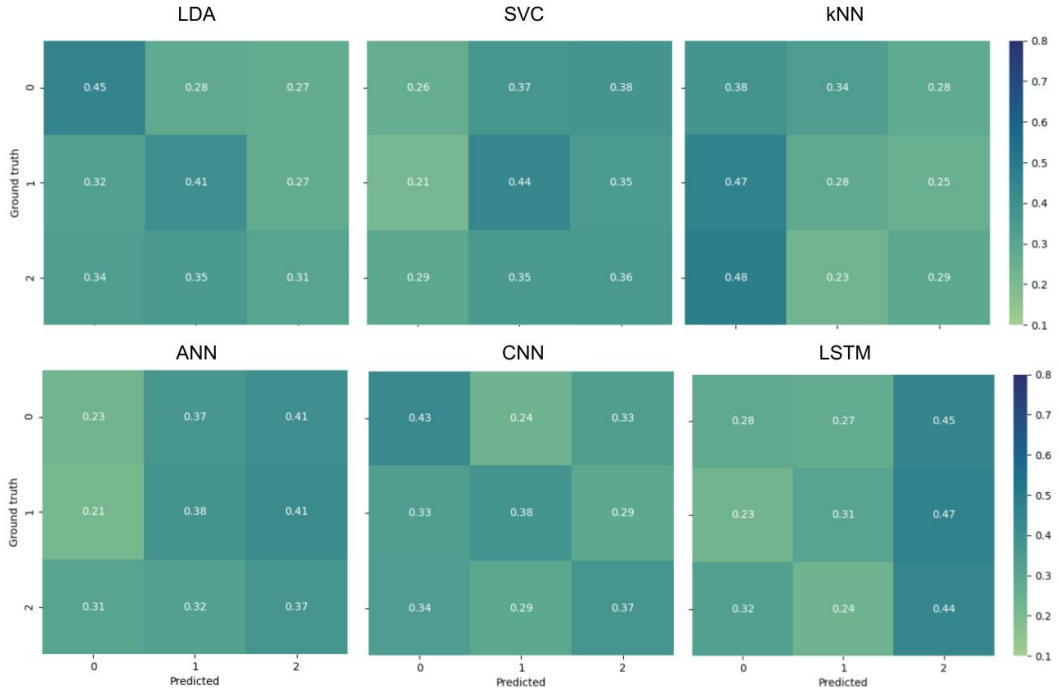

Figure 2: Confusion matrices for the subject-independent approach on the Shin et al. 2018 dataset of n-back tasks. Classes 0, 1 and 2 are 0-, 2- and 3-back respectively.

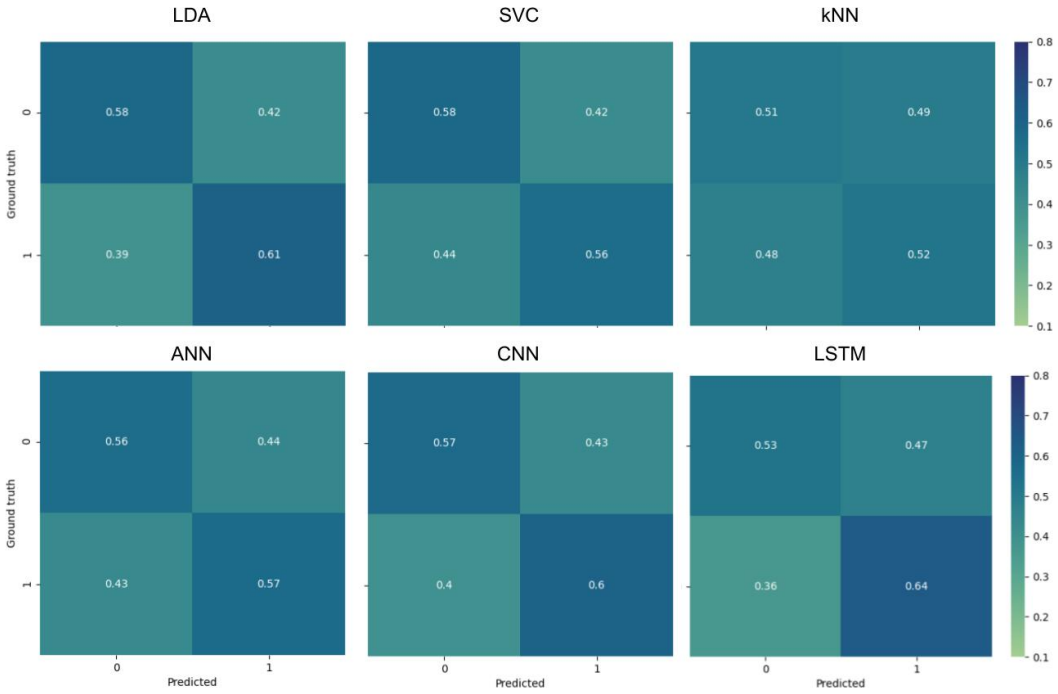

Figure 3: Confusion matrices for the subject-independent approach on the Shin et al. 2018 dataset of word generation tasks. Classes 0 and 1 are baseline task and word generation respectively.

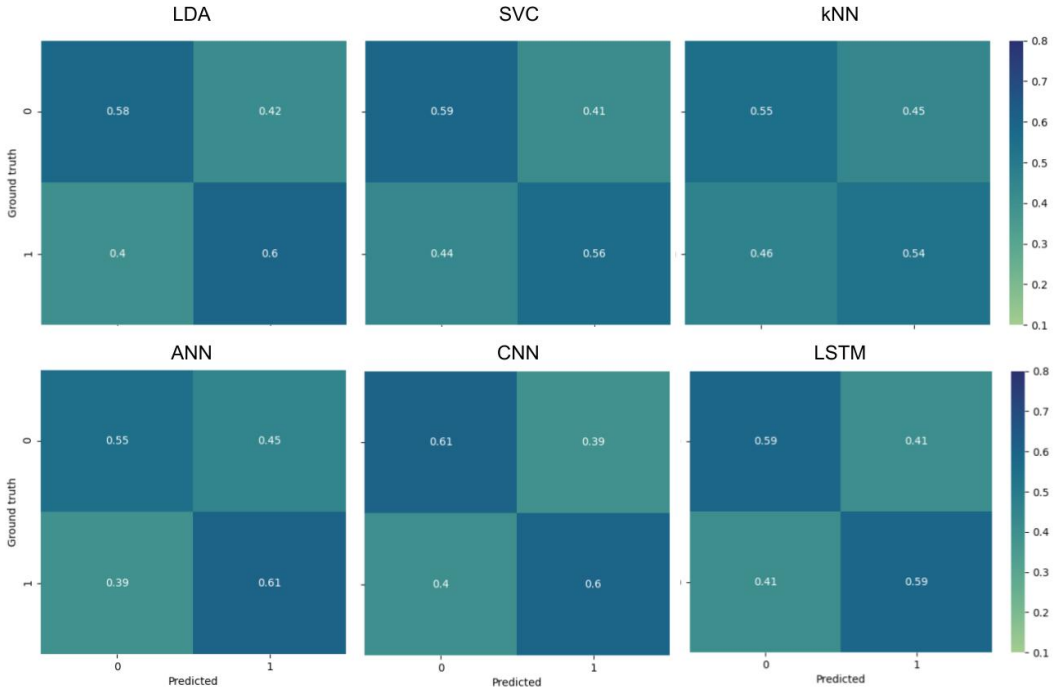

Figure 4: Confusion matrices for the subject-independent approach on the Shin et al. 2016 dataset of mental arithmetic tasks. Classes 0 and 1 are baseline task and mental arithmetic respectively.

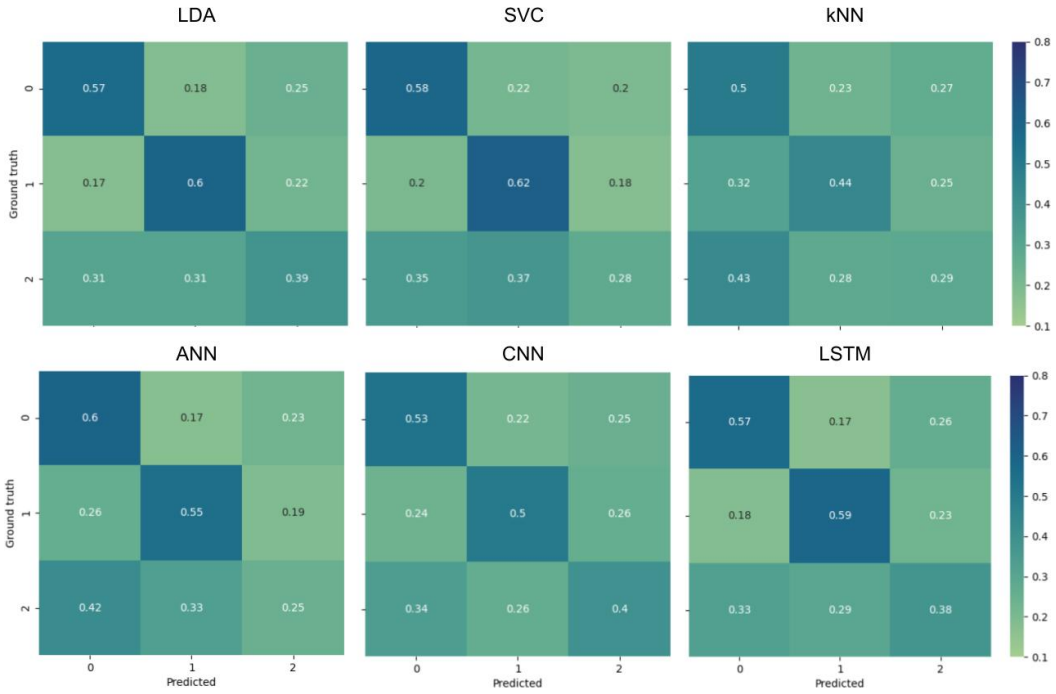

Figure 5: Confusion matrices for the subject-independent approach on the Bak et al. 2019 dataset of motor execution tasks. Classes 0, 1 and 2 are right hand finger tapping, left hand finger tapping and foot tapping respectively.

D Confusion matrices (Chapter 5)

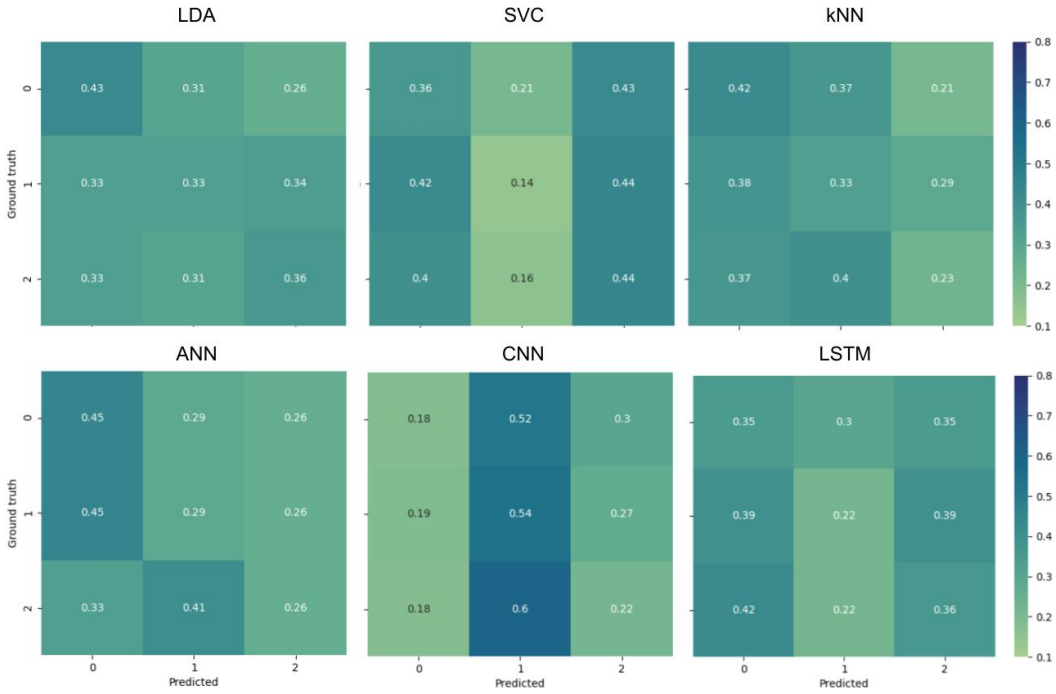

Figure 6: Confusion matrices for the n-back tailored subject-independent approach on the Herff et al. 2014 dataset of n-back tasks. Classes 0, 1 and 2 are 1-, 2- and 3-back respectively.

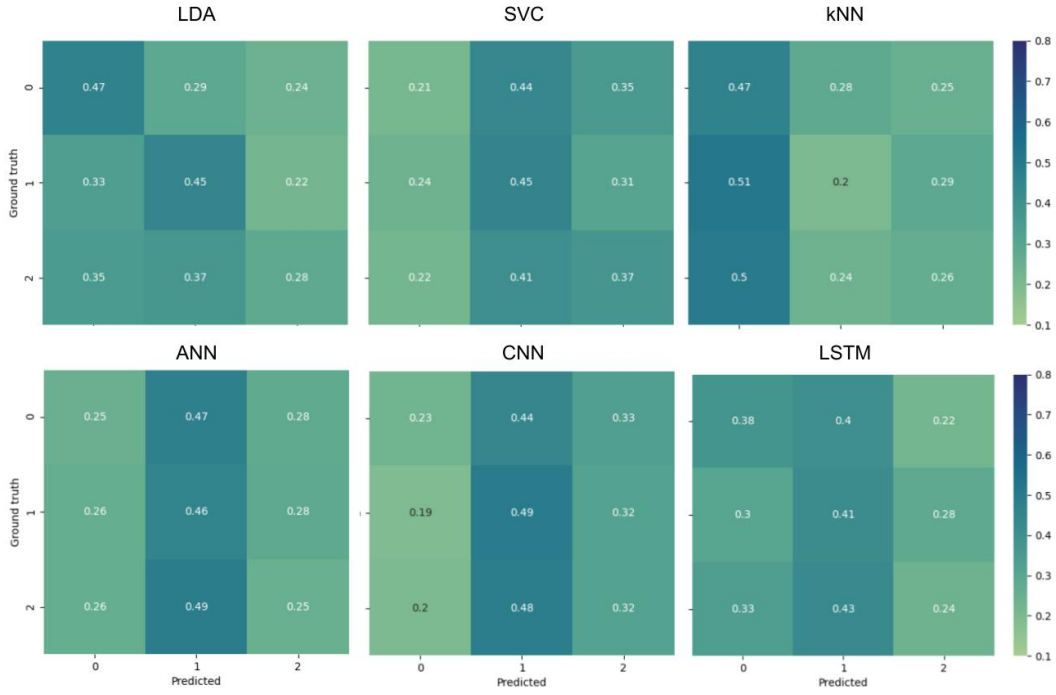

Figure 7: Confusion matrices for the n-back tailored subject-independent approach on the Shin et al. 2018 dataset of n-back tasks. Classes 0, 1 and 2 are 0-, 2- and 3-back respectively.