KU LEUVEN

FACULTY OF
ENGINEERING SCIENCE

Evaluation of distributed signal processing algorithms for online motor-imagery classification in wireless EEG sensor networks

Brent Sterckx

Thesis submitted for the degree of
Master of Science in
Electrical Engineering, option
Information Systems and Signal
Processing

Thesis supervisor:
Prof. dr. ir. Alexander Bertrand

Assessors:
Prof. dr. ir. Hugo van Hamme
Ir. Thomas Strypsteen

Mentor:
Ir. Cem Ates Musluoglu

Academic year 2021 – 2022

Preface

When I started work on this thesis, I was enthusiastic but at the same time worried it would become a year-long chore I would need to force myself to spend time on. I could not have imagined how wrong I was. The process of making this thesis and taking my first small steps into the world of scientific research has been an incredibly interesting and enriching one. I have learned a lot over the course of this year, both about the finer points of doing research as well as about myself.

This would not have been possible without all the people that supported me throughout this process. Therefore I would first and foremost like to thank my mentor Cem. You supported me in working independently but I could always count on you to help me look in the right direction when I got stuck and on your excellent guidance about research practices.
I am also grateful to my supervisor prof. Alexander Bertrand for allowing me to study this interesting topic in my thesis but also for introducing me to the incredibly interesting world of biomedical signal processing through your course on biomedical signal processing.

I also owe a great deal to my wonderful family and friends.
Words cannot describe how grateful I am for the unconditional love and support from my girlfriend Karen. Even though you had your own master thesis to complete, you were always there for me and listened to me rambling on and on about my thesis work.
Next, I am forever indebted to my parents for their continued support with everything I attempt and for enabling me to pursue my dreams.
I would also like to thank my classmates, who have over the years become dear friends, for all the fun moments that made the long hours of working on assignments more enjoyable. I could always exchange thoughts with you which has been incredibly valuable to me.

Finally I would also like to thank the members of the jury for taking the time to read this text as well as for attending the presentations.

Brent Sterckx

Contents

Abstract

Long-term, wearable EEG monitoring has the possibility to enable both new diagnostic tools as well as continuous, long-term BCI usage that could provide significant quality-of-life improvements for their users. However, current state-of-the-art methods of recording EEG are still ill-suited for long-term, mobile use. Therefore a novel approach was recently proposed that uses a wireless body area network of miniaturized EEG sensor nodes, together with distributed signal processing algorithms that efficiently use this novel architecture and which could be used in a BCI classification pipeline as a replacement of traditional feature extraction algorithms.

In this thesis, the performance and feasibility of these distributed algorithms as part of an end-to-end classification pipeline are evaluated for the application of motor imagery classification. To this end, literature on recently proposed distributed signal processing algorithms was combined to provide a complete set of feature algorithms that were subsequently integrated into different classification pipelines. Next, the performance of these pipelines was investigated based on the energy consumption, latency and accuracy by using a combination of both modelling as well as emulation of WESN architectures using previously recorded HD-EEG data.

As the optimal network size for this application was found to be relatively small, the distributed approaches performed worse than the naive centralization approach. Furthermore, re-use of a window of data for multiple iterations of the algorithm had no impact on the accuracy but efficient re-use had both lower latency and energy consumption as compared to non-efficient re-use. Increasing the number of channels per node can improve the accuracy for distributed pipelines at no extra cost in terms of energy consumption or latency. Furthermore, there is a maximum number of electrodes per node, dictated by energy and latency constraints. By designing for this maximum and with a constant battery size, a hardware implementation of a node could be designed that, through virtual channel selection, allows a single implementation to be tuned to individual subjects, without differences in performance w.r.t. latency or energy consumption.

In conclusion, distributed signal processing algorithms do not improve the performance of the BCI as the small network size reduces the amount of benefits gained from these algorithm to below the overhead they incur.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations and Symbols

Abbreviations

BCI	Brain-Computer Interface
MI	Motor Imagery
EEG	Electroencephalography
ERD	Event-Related Desynchronization
SMR	Sensorimotor Rhythm
CSP	Common Spatial Pattern
TRO	Trace-Ratio Optimization
MISO	Multiple-input-single-output
MIMO	Multiple-input-multiple-output
FBCSP	Filterbank CSP
GEVD	Genralized Eigenvector Decomposition
FIR	Finite Impulse Response
CSSP	Common Spatio-Spectral Patterns
LDA	Linear Discriminant Analysis
WESN	Wireless EEG Sensor Network
FCe	Fusion Center
FC	Fully Connected
TI	Topology Independent
AO	Alternate Optimization
RR	Round-robin
FIFO	First-in-first-out
NLOS	Non-line of sight
HD-EEG	High Definition EEG

Symbols

Throughout this text, scalars will be typeset as a small letter, vectors as a small bold letter and matrices as a bold capital letter. An exception to this are user-tunable parameters, which are typeset as capital letters.

N	Window length
L	Size of sliding window to compute $\mathbb{R}_l$ and $\mathbb{R}_r$
Q	Number of patterns
R	Re-use factor
α	Regularization parameter
$\mathcal{K}$	Set of nodes in the WESN
K	Size of the WESN
M_k	Number of electrodes of node k
M	Number of electrodes in the WESN
$\mathbf{x}_k[j]$	Observations of node k at time j

Chapter 1

Introduction

As a result of a spinal cord injury or neurological disease such as amyotrophic lateral sclerosis (ALS), patients may end up completely paralysed or even locked-in: fully awake but only able to move the eyes. Extensive research has been done on *brain-computer interfaces (BCI)* which aim to provide an interface between the human brain of a patient and a computer without using the normal neuromuscular outputs of the brain. Therefore, a BCI might enable these patients to once again interact with their surroundings, resulting in a substantial quality-of-life improvement. Indeed, this connection could be used for control of a physical device (e.g. steering an electric wheelchair or robotic arm) or communication device.

Also for able subjects, such an interface might provide a new way of interacting with a computer: for entertainment purposes for example. In the medical realm again, these subjects could potentially also benefit from a BCI as a diagnostic tool or for neuro-rehabilitation.

Ideally for these real-world applications, such a BCI device is hence wearable over longer periods of time such that the user is able to use the device at any time without requiring setup each time, which is especially relevant if the user requires help with this. In contrast to early research on BCI's which was typically constrained to short lab experiments, this text rather focusses on long-term, wearable neural monitoring.

1.1 Neurophysiological background

When designing brain-computer interfaces that pose to extract useful information directly from the brain, it is instructive to first take a quick look at how the brain works, how its activity can be measured, and what brain signals can be utilized. This neurophysiological background will then lead to algorithms that exploit these brain signals to extract useful information.

Information processing in the brain

The human brain is a network of brain cells or neurons, connecting neurons to one-another but also to the rest of the central nervous system [1]. While the interconnection of neurons in the brain itself provides the capability to process information, the connection to the rest central nervous system allows the brain to take in information from outside the brain and control the various processes in the body.

Communication between neurons happens in the form of action potentials [2]. Through either the release of a neurotransmitter, or through a current flow at the synapse, ion channels eventually open allowing ions to flow through the channel, disturbing the charge separation across the cell membrane. This reduces the membrane potential, termed depolarization. Indeed the influx of positively charged ions causes the interior of the neuron to become more positive whilst leaving the extracellular space more negative.
When the membrane is sufficiently depolarized and reaches a certain threshold, then, and only then, voltage-gated channels rapidly open, allowing a further influx of positively charged ions that further depolarize the neuron causing even more channels to open. The resulting electrical current caused by the flow of charged ions is called the *action potential*. It continues locally depolarizing the cell's membrane down the axon of the neuron towards the synapses, causing the action potential to run down the axon. At the synapses, the action potential can then elicit an action potential from a post-synaptic neuron. This is schematically illustrated in Fig. 1.1.
The threshold of the voltage-gated channels introduces a non-linearity in the neural signalling. This in fact creates the intelligence in the network.

Electroencephalography (EEG)

As neural signalling is based on the movement of charges, it creates currents and hence also a small electric fields. In *electroencephalography (EEG)*, electrodes are placed on the scalp to record these electric field potentials at the scalp, corresponding to neural signalling. These electrodes are usually connected to a fabric cap that the subject wears, that positions each electrode in a repeatable and known location. An example of a subject wearing such a cap is shown in Figure 1.2.

These signals measured at the scalp however, consist of a superposition of the individual signals from large groups of neurons located below an electrode owing to volume conduction rather than measuring the signals from a single neuron: the individual neurons' electric fields are propagated through the brain mass and are hence superimposed on top of one another.

Hence it should be noted that although brain signals originate from specific locations in the brain that handle a specific task, the resulting spatial resolution in the scalp EEG data is poor due to volume conduction [5]. The electrical fields created by

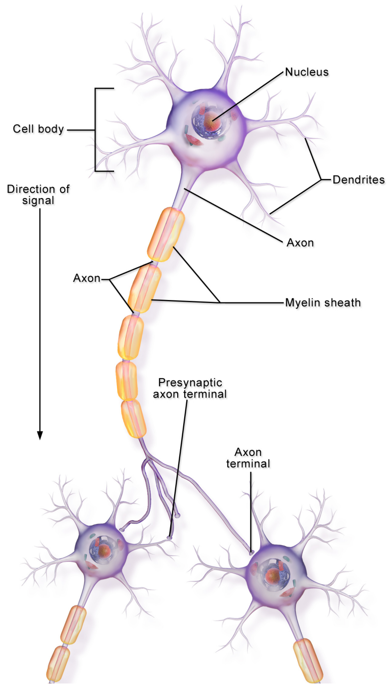


FIGURE 1.1: Schematic representation of connected neurons with the direction of the action potential running from the pre- to the post-synaptic neurons, indicated. Illustration by BruceBlaus, made available at [3] under a CC BY-SA 4.0 license

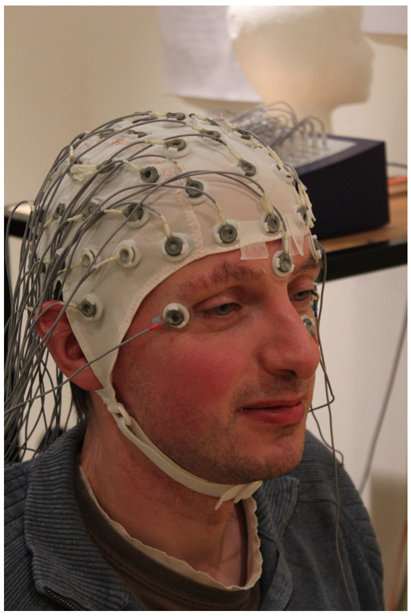


FIGURE 1.2: Example of a subject wearing an EEG electrode cap. Photo by Chris Hope, made available by Tim Sheerman-Chase at [4] through a CC BY 2.0 license

the neurons can propagate through the brain mass, resulting in a single spatially localized source being picked up by various neighbouring electrodes at the scalp. The electrode signals over a spread out area will be highly spatially correlated (under the assumption that the propagation to the scalp electrodes can be modelled as an instantaneous mixture).

In case a stronger signal originating from a different part of the brain is active in the same frequency band as a weaker signal of interest, the EEG measurements will be dominated by the first signal, making it impossible to obtain measurements of signals of the second signal. Typically, the occipital α-rhythm originating from the visual cortex, which is in the same band as the μ-rhythm, dominates EEG measurements in its band. In order to measure the μ-rhythm, spatial filtering is necessary to remove the interfering source, as will be described in Section 2.2. Note however that the statistics of EEG recordings are not stationary but slowly varying over time, requiring online adaptation to remain effective [6, 7, 8].

The low spatial resolution is a disadvantage of EEG. Much higher resolution signals

could be obtained using invasive techniques such as ECoG and neural probes which measure the electric fields respectively below the skull and from within the grey matter. Hence the effect of volume conduction is relatively lower as the signals from the targeted neuron group is much stronger now (propagated over a much smaller distance) as compared to the propagated signals of other neuron groups. In the same way this also allows to more locally measure the potentials and thus the spatial resolution is indeed increased.
However, their invasive nature makes them challenging for long-term use even aside from the inherent dangers of the required surgery. Although often experienced as uncomfortable, an EEG cap is non-invasive which avoids these challenges albeit at the cost of resolution. Moreover (and quite importantly for long-term neural monitoring as each patient will need to have its own system), an EEG system is relatively cheap compared to other clinical neurological research methods such as functional magnetic resonance imaging and magneto-encephalography.

Sensorimotor rhythms and event-related desynchronization

The action potential created by neurons' spiking is itself however too small to be detected by surface electrodes. On the other hand, as the brain is performing tasks, usually groups of thousands of neurons are involved. As their (de)polarization (cf. supra) displays a significant amount of synchrony, the superposition due to volume conduction (cf. supra) of the extracellular electrical fields – modulated by neural signalling – becomes measurable using EEG.

Different such groups of brain cells or regions handle specific tasks. Their activity is reflected in the EEG recordings in different ways [9], each leading to distinct brain signals that can be exploited. In MI-based BCI's, cortical oscillations produced by the sensorimotor cortices are currently the most used brain signal [10].

In relaxed awake subjects, the primary sensorimotor cortical areas produce rhythmic activity when they are not engaged in a task, generated by neurons in the respective cortex synchronously firing due to pacemaker cells or excitator/inhibitor loops [2]. These are called *cortical oscillations, sensorimotor rhythms (SMR), idle rhythms* or just *rhythms* [11, 5, 12]. The rhythms from different cortices can be distinguished from one-another by frequency, location over a certain cortex and/or relationship to concurrent sensory input or motor output since they are generated by different cortical structures which determines their dynamics.

In most adults, the sensorimotor cortices produce rhythmic activity between 8-12 Hz called the *(Rolandic)* μ*-rhythm.* Usually these rhythms are associated with 18-26 Hz β*-rhythms* but while some beta rhythms are harmonics of μ-rhythms, some are separable from them by topography and/or timing, and thus are independent EEG signals [13].

Upon movement or preparation of movement, a decrease of the μ- and β-rhythms' (band)power can usually be seen, especially contralateral to the brain region corresponding to the movement [11]. Since this occurs due to different neurons in the respective region desynchronizing their firing, this phenomenon is called *event-related desynchronization (ERD)*. Most useful for BCI's, this phenomenon is also seen for imagined or intended movement, referred to as *motor imagery (MI)*. This change in SMR characteristics enables the construction of an independent BCI based on MI by using the bandpower decrease due to ERD as a feature. Furthermore, it has been shown that paralysed subjects can learn to control SMR ERD, possibly providing a communication method for locked-in patients [9].

These two aspects make the SMR's ideal signals to serve as the basis for constructing a BCI. Therefore the classification pipeline used throughout this work will use SMR-based features that exploit the discriminative information provided by ERD.

1.2 Wireless EEG sensor networks

BCI's based on ERD of SMR's recorded using EEG have been extensively discussed in literature [5]. However, there are several problems with using the typical method of recording EEG for long-term neural monitoring, as is the focus of this work:

- Wearing an electrode cap for longer periods of time is uncomfortable;
- The wires connecting the electrodes to the processing system can lead to artifacts (e.g. when tugging or rubbing on the wires);
- The processing system is often bulky.

Therefore this work considers a different approach as proposed by Bertrand [14]. Herein the electrode cap is replaced by small battery-powered sensor nodes attached to the subject's head. They also contain a small processor and a wireless stack that lets them wirelessly interconnect to one another. A small amount of electrodes, in close proximity to the node, are also connected to each node. In each node, there is one electrode that acts as the (local) reference for the other electrodes of the node. This allows for miniaturization which increases user comfort: compared to an electrode cap, only small parts of the subject's scalp are covered by the nodes.
In [14], some architectures that could potentially exploit this are shown as examples. For example, a node could be realized as a small flexible system-on-chip that can be stuck to the users head. Another example is to implant either a small ECoG array or neural probes and having such a small node pick up and power the array or probes directly below it, thus avoiding the need for a transdural interface and as such increasing wearer comfort and safety.

Furthermore, as only specific regions of the brain are responsible for a certain task, nodes could be placed only over top of the regions of the scalp corresponding to that task[1]. This allows for even further miniaturization of the BCI, making the BCI more

[1]Due to volume conduction, the region at the scalp is always larger than the originating source

comfortable to wear and easier to set up.
In fact, it has been shown that for a MI BCI, the number of electrodes can be drastically lowered through optimal channel selection without decreasing performance [15]. The optimal electrode locations as determined by the channel selection objective are indeed extend over the brain regions that physiologically make sense for MI. Furthermore, this property extends to other applications as well, such as auditory attention detection [16, 17, 18].

In some cases, channel selection can even improve performance as task-irrelevant and redundant information is removed and as such the channel selection acts as a regularization method [17, 19]. The complexity of subsequent steps is also reduced.

1.3 Distributed signal processing

As discussed earlier, the spatial resolution of EEG is low due to the volume conduction effect and hence spatial filtering is typically used to overcome this problem. In a WESN, this will require centralizing the recordings from each of the nodes' electrodes in one of the nodes, or in a different device (that can also communicate wirelessly with the nodes) referred to as the *Fusion Center (FCe).*

The energy consumption required for this centralization however becomes prohibitively expensive as the capacity of the battery is shrunk due to miniaturization [14]. Furthermore, this naive approach to implementing a MI BCI on a WESN also doesn't utilize all of the available processing power; rather it only uses a single nodes'.

To overcome this problem, previous works have proposed efficient distributed realizations of popular optimal (spatial) filter problems to work in a WESN [20, 14, 21]. For example, a popular feature extraction algorithm for MI BCI's is common spatial pattern (CSP) analysis (see Section 2.2) which equates to the calculation of the GEVD of covariance matrices. A distributed realization for computing this GEVD was proposed by Bertrand and Moonen [20].

1.4 Goal of this thesis

Although the distributed realizations suggested in the previous section were proven to converge to the optimal solution of their central counterparts and their more efficient communication bandwidth requirements w.r.t. a naive centralization approach have been demonstrated, they have, to the best of the author's knowledge, not been evaluated in a complete acquisition-to-classification BCI implementation.

Compared to evaluating the distributed feature extraction in isolation, considering the complete classification pipeline has several advantages:

region. Electrodes should thus cover an extended patch at the scalp over top of the source to capture enough spatial information as to enable spatial filtering.

- Integration possibilities between the different components can be incorporated.
- The impact of design changes can be measured in terms of the output accuracy which is a more practically relevant metric than speed of convergence to the optimal solution (as is commonly studied for the distributed algorithms themselves, see for example [20, 14, 21]) as well as in more practical terms, e.g. in terms of battery life.
- Measures of the complete system can directly be related to system constraints (which are prohibitive for a naive approach [14]).

Hence such an evaluation would allow to more accurately assess the proposed approach w.r.t both performance as well as feasibility (w.r.t. hardware constraints). Further, it will allow for trade-offs in the parameter space to be discovered which equates to different operating points which will require manual selection rather than optimization w.r.t. the parameter. Therefore the goal of this thesis is to evaluate the distributed algorithms as integrated in a complete BCI, or more completely: to evaluate the use of distributed signal processing algorithms for online motor-imagery classification in wireless EEG sensor networks.

The evaluation will be based on three objectives drawing from both the performance and feasibility aspects: energy consumption, latency and decoding accuracy.

Energy consumption

As the nodes envisioned are to be small such that they can be comfortably worn on the user's head for longer periods of time, the maximal battery capacity is also limited and the system is energy constrained. For ease of use however, the battery of a node should last sufficiently long. As explained earlier, energy consumption prohibits fusing all the signals in a single node and straightforwardly applying conventional feature extraction algorithms. Therefore, an important goal in designing the envisioned BCI with distributed algorithms, is to minimize energy consumption to increase battery life.

> **Objective 1.** Minimal energy consumption

Latency

For the BCI to feel responsive or to be able to control a device with sufficient time-fidelity, the input-output latency should be minimal.

> **Objective 2.** Minimal latency

In an asynchronous system, i.e., a system in which the user can choose when to give or not give commands at his or her own pace, the *input-output latency* or also just *latency* of the system is defined as follows:

Definition 1.1. The (input-output) latency of an asynchronous system is the time it takes for a command given to the system to have an effect on the decoded output.

As in this work the simplification is made towards a synchronous system, i.e., the user controls the output by responding to cues given by the system at regular time intervals, this definition must be altered:

Definition 1.2. The (input-output) latency in a synchronous system is the time it takes from the first input sample after the cue has been presented to have an effect on the decoded output.

Furthermore, the processing done after the output is given needs to be sufficiently fast such that it is complete before the next inputs must be processed. Therefore, minimal latency is not only an objective but also a requirement for feasibility: only a certain maximum of total latency can be tolerated.

Decoding accuracy

Finally the BCI should be as accurate as possible in decoding the correct class. This will make it more attractive to potential users since it increases the information transfer rate (ITR) that can be achieved and avoids the frustration that comes along with misinterpretation of commands.

> **Objective 3.** Maximal decoding accuracy

Although the accuracy will be included as a metric, this text's focus is on the distributed implementation and as such, achieving maximum accuracy will not be the primary goal. Rather, the accuracy will be used to compare implementation to each other.

1.5 Contents

The rest of this text proceeds in two parts.

Part I concerns the methods used for MI methods used for MI classification.
In *Chapter 2*, the architecture of a typical MI-based BCI and the details of the selected algorithms are presented. This typical architecture will then serve as the framework for adapting to run on a WESN in the next chapter. In *Chapter 3*, distributed versions of a portion of the typical architecture will be derived for different WESN topologies. This chapter then concludes with a description on how to integrate those distributed algorithms within the classification framework of the previous chapter in order to (efficiently) do distributed, online MI classification on a WESN. This provides a complete description of the proposed distributed classification system.

After describing the distributed BCI, *Part II* of this thesis focuses on evaluating the performance and feasibility of the proposed approach w.r.t. the objectives defined in Section 1.4. To do so, *Chapter 4* first presents models for each of those objectives for the distributed algorithms. Next, *Chapter 5* presents the actual evaluation of the distributed algorithms based on offline simulations on experimental data. Furthermore, the feasibility is assessed w.r.t. the current state-of-the-art of the various components.

Finally *Chapter 6* gives some final conclusions on the text as well as some considerations for future work.

Part I

Algorithms

Chapter 2

MI classification

Before presenting distributed MI classification within a WESN, it is instructive to look at a typical approach to the problem of classifying (imagined) left and right hand movement to serve as a basis for distributed classification. The prototypical pipeline from data acquisition to classifier output is schematically represented in Figure 2.1.

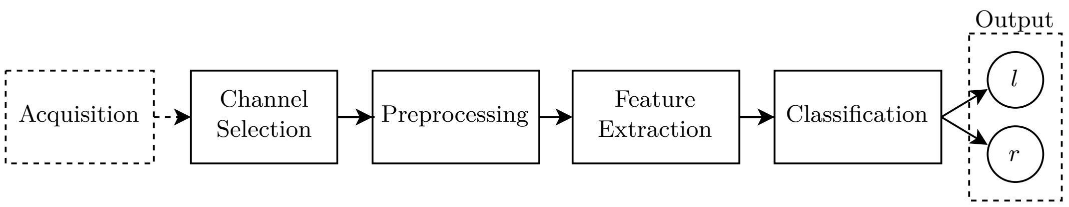


FIGURE 2.1: Prototypical motor-imagery classification pipeline

Typically the first step in the pipeline is a *channel selection* procedure to select the channels that are most relevant to the classification. This has multiple advantages:

- It reduces the computational burden;
- It reduces overfitting;
- It removes the noise that is present in the channels that do not contribute to the objective;
- It reduces acquisition setup time;
- It increases wearer comfort.

The fact that some channels will not contribute to the objective, is demonstrated in Figure 2.2. Here the average power-spectral densities are shown for a subject engaged in a MI task. Channels located over the locations in the sensorimotor cortex responsible for processing of hand movements (e.g. `C3` and `C4`) show a difference in the μ-band while channels further away are not discriminative (compare `C3` and `C4` with `F1` and `F2`). In the former group of channels, a lateralization due to ERD is also visible: the power in the μ-band is decreased contralaterally.
These observations are in accordance with the neurophysiology as discussed in the

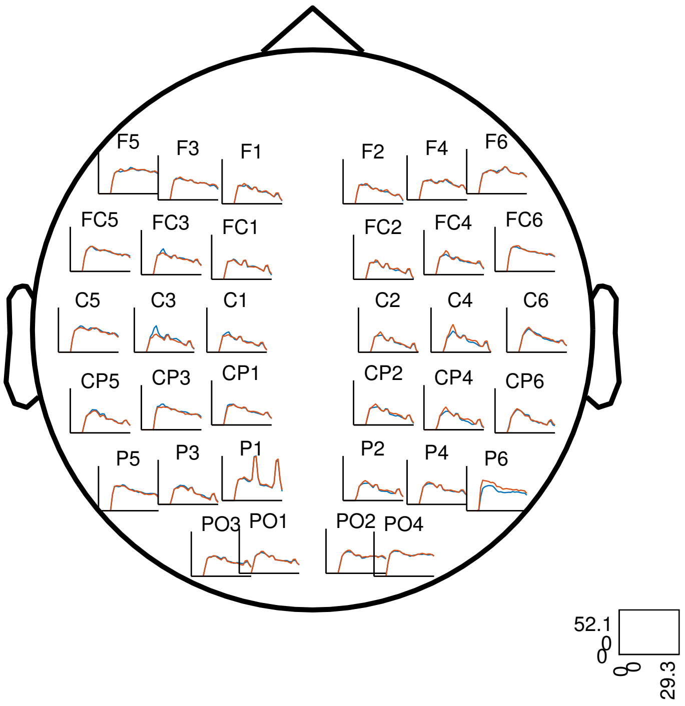


FIGURE 2.2: Averaged power-spectral densities of the different channels for both conditions(blue=`l`, orange=`r`)

previous chapter.

Channel selection can either be done before acquisition – only the relevant electrodes are placed and recorded – or after.
In the former case, the selection is done from neuroanatomical insights or from an earlier post-experiment channel selection (preferably from the same subject).
In the latter case, typically a measure for the performance of the resulting classifier is used to select the optimal subset of channels [17, 16, 18].

As the channel selection method will need to be extended to a node selection method in the next chapter, and no such technique exists that is immediately useable for the application at hand[1], channel selection methods will not be discussed further in this work and the nodes will be selected manually, see Chapter 5.

Next, the remaining channels are *preprocessed* to remove the irrelevant parts. This increases the quality of the extracted features in the next step: *feature extraction*.

[1]Roadblocks are the non-linear classifier and nodes should not overlap, hence a typical grouped selection approach is insufficient

The feature extraction poses to extract compact measures about the data. To be useful for the task of classifying into two classes, these features must be discriminative between the two classes. For a MI BCI, the features are typically determined per window of data such that it would allow to have real-time control over an application (e.g. an electric wheelchair).
In the final step, *classification*, the classifier then uses these features to classify into one of two classes, which we will further denote as l and r for respectively left and right (imagined) hand movement.

The rest of this chapter details the motivation behind the selected method of each of the steps of the classification pipeline as well as some of their specifics.

2.1 Preprocessing

As the SMR's are the useful part of the EEG recordings for the proposed BCI (see Section 1.1), the goal is to only extract these signals from the raw measured data. By only keeping the information related to these signals, the quality of the extracted features is improved since the noise from the non-useful part of the data is removed.

Since the μ-band is known to be active between 8-12 Hz and the β-band between 18-26 Hz [9], the other frequency info can be removed by bandpass filtering all the EEG channels with a 8-26 Hz passband. As ERD and ERS are frequency domain features – they are not phase-locked – the precise timing is not required. Since the signal after filtering only has content up to 26 Hz, it can hence be downsampled without losing information. This lowers the storage and computational requirements in the rest of the pipeline.

Since the feature extraction algorithms to be described further on in Section 2.2 are invariant to linear transformations, re-referencing the data is also not needed either [22].

Furthermore, since those algorithms only rely on statistical information from the signals that is useful in discriminating the output classes, artifacts are implicitly suppressed and hence artifact removal is not necessary. Note that this is only the case if artifacts are equally distributed in both output conditions [5, p. 50].

It should be noted that both frequency filtering and downsampling can efficiently be implemented directly in hardware to run consecutively with the sampling. In this way, the preprocessing incurs almost no additional latency and the system requires less memory.

2.2 Feature extraction – Common Spatial Patterns

Due to volume conduction, the spatial resolution of a cortex' signals across the scalp is known to be poor (see Section 1.1). For example, electrodes placed in the region of the sensorimotor cortex will pick up a.o. the much stronger occipital α-rhythm and movement and muscle artifacts such that the useful μ- and β-rhythms have a low SNR in most of the electrodes on the scalp. Consequently, the modulation of the power of those bands due to ERD and ERS extracted from only a single electrodes would result in poor classification performance.
However, by combining signals from multiple electrodes, the spatial information can be used and the useful signals' SNR can be increased. This is commonly referred to as *spatial filtering* (rather than spectral filtering which would filter w.r.t. time/frequency of time).

Actually for classification, more important than just increasing the ratio of the useful rhythms' power to the noise power for one of the classes, is spatially filtering the channels in such a way that the classes to be classified $\{l, r\}$ are best separable in some sense.
A popular technique in MI BCI literature is *Common Spatial Pattern (CSP) analysis* [23, 24]. CSP analysis looks for the linear spatial filter $\mathbf{w}^*$ that optimizes the ratio of the mean power of the two classes [5, 18]. Hence the modulation of the output power of this filter (due to modulation of the SMR's caused by ERD/ERS present in the channels) will be maximally discriminative between the two classes. Furthermore, by computing such a filter for each of the classes, the ratio of the filter output powers is also maximally discriminative between the two classes, allowing classification based on these output powers, as will be explained below.

For example if we want to extract class $\{l, r\}$ while suppressing $\{r, l\}$, the objective is formulated as follows:

$$\mathbf{w}^*_{\{l,r\}} = \arg\max_{\mathbf{w}} \frac{E\left[\left(\mathbf{w}^\top \mathbf{x}_{\{l,r\}}\right)^2\right]}{E\left[\left(\mathbf{w}^\top \mathbf{x}_{\{r,l\}}\right)^2\right]} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^\top \mathbf{R}_{\{l,r\}} \mathbf{w}}{\mathbf{w}^\top \mathbf{R}_{\{r,l\}} \mathbf{w}}\,, \tag{2.1}$$

where the subscripts must be chosen such that each variable's subscript is in the same position in the set given in that variable's subscript in the equation above. $\mathbf{x}_{\{l,r\}}$ denotes observations from class $\{l, r\}$, $\mathbf{R}_{\{l,r\}}$ denotes the covariance matrix of the observations corresponding to class $\{l, r\}$. $\mathbf{w}^*_l$ then gives the filter that optimally increases the mean power of class l w.r.t. the mean power of class r and vice versa for $\mathbf{w}^*_r$.

In the next subsection, it will be proven that the solution for both $\mathbf{w}^*_l$ and $\mathbf{w}^*_r$ can be found by computing a single GEVD of $\left(\mathbf{R}_{\{l,r\}}, \mathbf{R}_{\{r,l\}}\right)$ and setting $\mathbf{w}^*_l = \mathbf{w}_1$ and $\mathbf{w}^*_r = \mathbf{w}_M$, where $\mathbf{w}_1$ and $\mathbf{w}_M$ are respectively the generalized eigenvectors with the largest and smallest generalized eigenvalues.

In practice however, the covariance matrices are not available. Therefore, they have to be estimated based on labelled, pre-recorded data. For a window of $N_{\{l,r\}}$ observations of class $\{l, r\}$, the corresponding covariance matrix can be estimated as

$$\mathbf{R}_{\{l,r\}} \approx \frac{1}{N_{\{l,r\}}} \sum_{\mathbf{x}_i \in \{l,r\}} \mathbf{x}_i \mathbf{x}_i^\top , \tag{2.2}$$

where $\mathbf{x}_i \in \{l, r\}$ denotes running over all $n_{\{l,r\}}$ observations of class $\{l, r\}$. Because of the need of labelled data, CSP analysis is a supervised method.

Adaptation

The filters extracted this way are however unsuitable for long-term use as is the intent of the scheme proposed in this work. This is because the statistics used by CSP, as captured by the covariance matrices, slowly change over time as e.g.; the subject gets more fatigued or loses focus. These changes are noticeable within the duration of 1 typical experiment (< 1 day) but degrade performance as the duration increases – with long-term use even to virtually infinitely long [6, 7, 8]. Therefore, to enable long-term use, the feature extraction should adapt to the slowly varying statistics.

A simple solution to make CSP adaptive, would be to recompute the covariance matrix for each class and recompute the filter matrix at the end of each window, using a sliding window of previous windows of observations. However, as in the application considered in this text, observations in a window are considered to be of only one class, this window should be taken sufficiently large, expressed as the number of included windows L, such that it can be assumed that both classes are included. Another solution is to recompute the optimal filters after each window using a fixed amount of previous windows of data, hence sliding a second, larger window over the data such that there are a fixed amount of previous windows available to calculate the covariance matrices, assuming both classes are included. However in a real-world scenario, the label of that window is not available. Therefore, labelling of past windows is based on for example the predicted class instead. As long as the classifier is sufficiently accurate originally, the update procedure will then keep it that way as the statistics change.

In this text however, the problem will be simplified by assuming the true label of the window is available. In practice this assumption is of course not valid, but a classifier with at least some degree of accuracy (i.e. $> 50\%$) can approximate this assumption arbitrarily well by having the user repeat the instruction multiple times, classifying these trials and using a voting system, rather than doing single-trial classification. Studying the impact of a non-perfect classification on the updating procedure is outside the scope of this text.

Features

Finally, the features are taken to be the average power of the filters' outputs for both classes (i.e., of $\mathbf{w}_l^*$ and $\mathbf{w}_r^*$) over a window of size N and next converting them into the log-domain. These features capture the modulation of the SMR's power due to ERD and ERS, with optimal discrimination between the two classes for $N \to \infty$. Indeed the average filter output power is an unbiased estimator of the mean filter output power which is precisely what the CSP method has made optimally discriminative. Note that the window over which one feature vector is calculated can be taken to be the same window as used for updating the covariance matrices.

In the next subsection, it will be shown that the solution to the optimization problem central to the CSP method (2.1) is found by computing the GEVD of the covariance matrices. This derivation will then straightforwardly allow to extend the method to:

- using multiple filters per class (Subsection 2.2.2),
- combined spatial and temporal filtering (Subsection 2.2.3),
- incorporating regularization (Subsection 2.2.4).

2.2.1 Solution to the (MISO) CSP optimization problem

In this subsection, the solution to the CSP optimization problem (2.1) is derived as it will be instructive for generalizing and adapting the CSP method in respectively the next subsections and next chapter. Further, it provides insight into the meaning of the solution.

The solution of (2.1) is only unique up to scaling, as multiplying $\mathbf{w}$ with a scaling factor in both numerator and denominator can easily be seen to cause it to cancel out in the fraction, resulting in the same value for the objective. Therefore, the scaling can be arbitrarily fixed, up to the sign, by adding the constraint $\mathbf{w}^\top \mathbf{R}_{\{r,l\}} \mathbf{w} = 1$:

$$\begin{aligned} \mathbf{w}^*_{\{l,r\}} = \arg\max_{\mathbf{w}} \; & \frac{\mathbf{w}^\top \mathbf{R}_{\{l,r\}} \mathbf{w}}{\mathbf{w}^\top \mathbf{R}_{\{r,l\}} \mathbf{w}} \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{R}_{\{r,l\}} \mathbf{w} = 1\,, \end{aligned} \tag{2.3}$$

$$\begin{aligned} = \arg\max_{\mathbf{w}} \; & \mathbf{w}^\top \mathbf{R}_{\{l,r\}} \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{R}_{\{r,l\}} \mathbf{w} = 1\,. \end{aligned} \tag{2.4}$$

Using the theory of Lagrange multipliers for convex problems, we can then solve the constrained problem as follows. The Lagrangian is given by

$$\mathcal{L}_{\{l,r\}}(\mathbf{w}, \lambda) = \mathbf{w}^\top \mathbf{R}_{\{l,r\}} \mathbf{w} + \lambda \left(\mathbf{w}^\top \mathbf{R}_{\{r,l\}} \mathbf{w} - 1 \right) \tag{2.5}$$

and the optimum is given by

$$\nabla_{(\mathbf{w},\lambda)}\mathcal{L}_{\{l,r\}}(\mathbf{w}^*,\lambda^*) = 0 \tag{2.6}$$

$$\leftrightarrow \begin{cases} \mathbf{R}_{\{l,r\}}\mathbf{w}^* = \lambda^*\mathbf{R}_{\{r,l\}}\mathbf{w}^* \\ \mathbf{w}^{*\top}\mathbf{R}_{\{r,l\}}\mathbf{w}^* = 1\,. \end{cases} \tag{2.7}$$

Note that equation (2.7) is a GEVD of the pair $\left(\mathbf{R}_{\{l,r\}}, \mathbf{R}_{\{r,l\}}\right)$. This form reveals why this method is called CSP.
The GEVD computes a matrix $\mathbf{W} = \left[\begin{array}{cccc} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_M \end{array}\right]$ that *jointly* diagonalizes the matrices $\mathbf{R}_{\{l,r\}}$ and $\mathbf{R}_{\{r,l\}}$ into the diagonal matrices $\left(\mathbf{\Sigma}_{\{l,r\}}, \mathbf{\Sigma}_{\{r,l\}}\right)$:

$$\begin{cases} \mathbf{W}^\top\mathbf{R}_{\{l,r\}}\mathbf{W} = \mathbf{\Sigma}_{\{l,r\}} \\ \mathbf{W}^\top\mathbf{R}_{\{r,l\}}\mathbf{W} = \mathbf{\Sigma}_{\{r,l\}} \end{cases}, \tag{2.8}$$

$$\text{where} \quad \begin{array}{l} \mathbf{\Sigma}_{\{l,r\}} = \operatorname{diag}(\lambda_{\{l,r\},1}, \lambda_{\{l,r\},2}, \dots, \lambda_{\{l,r\},M}) \\ \mathbf{\Sigma}_{\{r,l\}} = \operatorname{diag}(\lambda_{\{r,l\},1}, \lambda_{\{r,l\},2}, \dots, \lambda_{\{r,l\},M}) \end{array} \tag{2.9}$$

and where diag$(\cdot)$ denotes the diagonal matrix with the arguments as the diagonal elements. Next, w.l.o.g. we can take

$$\lambda_{\{l,r\},1} \geq \lambda_{\{l,r\},2} \geq \dots \geq \lambda_{\{l,r\},M}\,. \tag{2.10}$$

Furthermore, $\mathbf{W}$ is once again unique up to scaling. Indeed, scale $\mathbf{w}_i$ with factor a_i. Then the matrix $\mathbf{W}$ is scaled as $\mathbf{W} \cdot \operatorname{diag}(a_1, \dots, a_M)$, such that (2.8) still holds but with generalized eigenvalues that are scaled with factor a_i^2.
Hence the generalized eigenvalues of $\mathbf{R}_{\{r,l\}}$ can be set to 1 to adhere to the constraint in (2.7):

$$\lambda_{\{r,l\},i} = 1 \quad \forall i \in \{1, 2, \dots, M\}\,. \tag{2.11}$$

The original objective of (2.1) can then be rewritten i.f.o. this joint diagonalization as follows: Take $\mathbf{w}_i$ to be a generalized eigenvector as in (2.8), then

$$\frac{\mathbf{w}_i^\top\mathbf{R}_{\{l,r\}}\mathbf{w}_i}{\mathbf{w}_i^\top\mathbf{R}_{\{r,l\}}\mathbf{w}_i} = \frac{\lambda_{\{l,r\},i}}{\lambda_{\{r,l\},i}} = \lambda_{\{l,r\},i}\,. \tag{2.12}$$

This can then be interpreted as follows. Clearly from the above equation, by taking $\mathbf{w}^*_{\{l,r\}} = \mathbf{w}_1$, the objective is maximized (and minimized for $\mathbf{w}^*_{\{l,r\}} = \mathbf{w}_M$). This equates to selecting the spatial filter or spatial pattern that decomposes the class covariance matrices in the 'direction' of maximal power ratio. In this case due to the meaning of the elements of $\mathbf{w}_i$, the 'direction' is to be interpreted as a spatial pattern/spatial filter. Because it boosts one class' characteristics and at the same time attenuates the other class' characteristics, the filter is 'common' between the two classes.

Further, note that by replacing the maximization with a minimization in (2.1), $\mathbf{w}^*_{\{r,l\}}$ is found. This can be seen by inverting the problem for $\mathbf{w}^*_{\{r,l\}}$. This puts the covariance matrices in the same place as for $\mathbf{w}^*_{\{r,l\}}$ and the maximization becomes a minimization. Hence from a single GEVD, maximally discriminative filters for both classes are found.

Applying CSP analysis for feature extraction thus boils down to computing GEVD $\left(\mathbf{R}_{\{l,r\}}, \mathbf{R}_{\{r,l\}}\right)$ and setting $\mathbf{w}^*_l = \mathbf{w}_1$ and $\mathbf{w}^*_r = \mathbf{w}_M$.

It is now also easily proven that the ratio of the power of the output of the filters $\mathbf{w}^*_l$ and $\mathbf{w}^*_r$ is maximal when given observations of one of the classes:

$$\frac{E\left[\left(\mathbf{w}_l^{*\top}\mathbf{u}_l\right)^2\right]}{E\left[\left(\mathbf{w}_r^{*\top}\mathbf{u}_l\right)^2\right]} = \frac{\mathbf{w}_l^{*\top}\mathbf{R}_{\{l,r\}}\mathbf{w}_r^*}{\mathbf{w}_r^{*\top}\mathbf{R}_{\{l,r\}}\mathbf{w}_r^*} = \frac{\lambda_{\{l,r\},1}}{\lambda_{\{l,r\},M}} \,. \tag{2.13}$$

2.2.2 MIMO

So far, only a single filter/direction was extracted per class, i.e. each filters represents a MISO system. It is however reasonable that the relevant subspace is multi-dimensional.

To this end, the objective of eq. (2.1) can be generalized to extract multiple filters per class, i.e. a MIMO system [25, 26, 27, 28]. To extract $Q/2$ filters $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{Q/2}$ of class $\{r, l\}$, define the filter matrix $\mathbf{W} = \left[\begin{array}{cccc} \mathbf{w}_1 & \mathbf{w}_2 & \ldots & \mathbf{w}_{Q/2} \end{array}\right] \in \mathbb{R}^{M\times Q/2}$. Then the objective becomes

$$\mathbf{W}^*_{\{l,r\}} = \underset{\mathbf{W}}{\arg\max}\, \frac{E\left[\left\|\mathbf{W}^\top\mathbf{x}_{\{l,r\}}\right\|^2\right]}{E\left[\left\|\mathbf{W}^\top\mathbf{x}_{\{r,l\}}\right\|^2\right]} = \underset{\mathbf{W}}{\arg\max}\, \frac{\mathrm{Tr}\left(\mathbf{W}^\top\mathbf{R}_{\{l,r\}}\mathbf{W}\right)}{\mathrm{Tr}\left(\mathbf{W}^\top\mathbf{R}_{\{r,l\}}\mathbf{W}\right)} \,, \tag{2.14}$$

where $\mathbf{W}^*_{\{l,r\}} = \left[\begin{array}{cccc} \mathbf{w}^*_1 & \mathbf{w}^*_2 & \ldots & \mathbf{w}^*_{Q/2} \end{array}\right]$ is a $\mathbb{R}^{M\times Q/2}$ matrix containing the $Q/2$ optimal spatial filters. Then, to fully constrain $\mathbf{W}$, the constraint $\mathbf{W}^\top\mathbf{W} = \mathbb{I}$ is added.

The optimization problem above is commonly referred to as the *Trace Ratio Optimization (TRO)* problem. However, different from the MISO case, the TRO problem does not have a closed-form solution. Therefore, the constraint is usually replaced by $\mathbf{W}^\top\mathbf{R}_{\{r,l\}}\mathbf{W} = \mathbb{I}$ which is again a generalization of the MISO case. This leads to the simpler, yet not equivalent problem:

$$\begin{aligned} \mathbf{W}^*_{\{l,r\}} = \underset{\mathbf{W}}{\arg\max}\ & \mathrm{Tr}\left(\mathbf{W}^\top\mathbf{R}_{\{l,r\}}\mathbf{W}\right) \\ \text{s.t.}\quad & \mathbf{W}^\top\mathbf{R}_{\{r,l\}}\mathbf{W} = \mathbb{I} \,. \end{aligned} \tag{2.15}$$

The Lagrangian of this problem is given by

$$\mathcal{L}(\mathbf{W}, \mathbf{\Lambda}) = \operatorname{Tr}\left(\mathbf{W}^\top \mathbf{R}_{\{l,r\}} \mathbf{W}\right) - \operatorname{Tr}\left(\left(\mathbf{W}^\top \mathbf{R}_{\{r,l\}} \mathbf{W} - \mathbb{I}\right) \mathbf{\Lambda}\right) , \tag{2.16}$$

with $\mathbf{\Lambda}$ a $\mathbb{R}^{Q/2 \times Q/2}$ (not necessarily diagonal) matrix of Lagrange multipliers.

The solution to this convex problem is given by the following equation according to the theory of Lagrange multipliers [22]:

$$\nabla_{(\mathbf{W},\mathbf{\Lambda})} \mathcal{L}_{\{l,r\}}(\mathbf{W}^*_{\{l,r\}}, \mathbf{\Lambda}^*_{\{l,r\}}) = 0 \tag{2.17}$$

$$\leftrightarrow \begin{cases} \mathbf{R}_{\{l,r\}} \mathbf{W}^*_{\{l,r\}} = \mathbf{R}_{\{r,l\}} \mathbf{W}^*_{\{l,r\}} \mathbf{\Lambda}^*_{\{l,r\}} \\ {\mathbf{W}^*_{\{l,r\}}}^\top \mathbf{R}_{\{r,l\}} \mathbf{W}^*_{\{l,r\}} = \mathbb{I} \end{cases} . \tag{2.18}$$

Now multiplying the first equation with $\mathbf{W}^*_{\{l,r\}}$ and combining with the second equation of (2.18) gives $\mathbf{\Lambda}^* = {\mathbf{W}^*_{\{l,r\}}}^\top \mathbf{R}_{\{l,r\}} \mathbf{W}^*$. This shows that $\mathbf{\Lambda}^*_{\{l,r\}}$ is symmetric and thus an orthonormal diagonal decomposition $\mathbf{\Lambda}^*_{\{l,r\}} = \mathbf{U}_{\{l,r\}} \mathbf{\Sigma}^*_{\{l,r\}} \mathbf{U}^\top_{\{l,r\}}$ exists. Thus the first equation of (2.18) can be rewritten as

$$\mathbf{R}_{\{l,r\}} \mathbf{W}^*_{\{l,r\}} = \mathbf{R}_{\{r,l\}} \underbrace{\mathbf{W}^*_{\{l,r\}} \mathbf{U}_{\{l,r\}}}_{\triangleq \mathbf{V}^*_{\{l,r\}}} \mathbf{\Sigma}^*_{\{l,r\}} \mathbf{U}^\top_{\{l,r\}} \tag{2.19}$$

$$\leftrightarrow \mathbf{R}_{\{l,r\}} \mathbf{V}^*_{\{l,r\}} = \mathbf{R}_{\{r,l\}} \mathbf{V}^*_{\{l,r\}} \mathbf{\Sigma}^*_{\{l,r\}} . \tag{2.20}$$

Note that as in the MISO case (eq. (2.7)), this leads to the GEVD of the pair $\left(\mathbf{R}_{\{l,r\}}, \mathbf{R}_{\{r,l\}}\right)$ where $\mathbf{V}^*_{\{l,r\}} = \left[\begin{array}{cccc} \mathbf{v}^*_{\{l,r\},1} & \mathbf{v}^*_{\{l,r\},2} & \cdots & \mathbf{v}^*_{\{l,r\},Q/2} \end{array}\right]$ contains $Q/2$ generalized eigenvectors corresponding to the generalized eigenvalues $\mathbf{v}^*_i$ (not necessarily the largest) on the diagonal of $\mathbf{\Sigma}^*_{\{l,r\}}$, i.e. $\mathbf{\Sigma}^*_{\{l,r\}} = \operatorname{diag}(\mathbf{v}^*_{\{l,r\},2}, \mathbf{v}^*_{\{l,r\},2}, \ldots, \mathbf{v}^*_{\{l,r\},Q/2})$. Per definition of $\mathbf{V}^*_{\{l,r\}}$, the spatial filters $\mathbf{W}^*_{\{l,r\}}$ are a unitary transformation of the generalized eigenvectors contained in $\mathbf{V}^*_{\{l,r\}}$ and thus lie in the subspace spanned by the latter.

Now consider the joint diagonalization given by the GEVD as in eqs. (2.8-2.10) with $\mathbf{V}$ instead of $\mathbf{W}$ for notational clarity. Using the constraint ${\mathbf{W}^*_{\{l,r\}}}^\top \mathbf{R}_{\{r,l\}} \mathbf{W}^*_{\{l,r\}} = \mathbb{I}$, again fixes the scaling of the eigenvectors as in (2.11).

Similarly to the MISO case (eq. (2.12)) this joint diagonalization given by the GEVD can then be used to rewrite the optimal value of the objective as given in eq. (2.15) i.f.o. the generalized eigenvalues:

$$\begin{aligned}
&\operatorname{Tr}\left(\mathbf{W}^{*}_{\{l,r\}}{}^{\top}\mathbf{R}_{\{l,r\}}\mathbf{W}^{*}_{\{l,r\}}\right) && (2.21)\\
&= \operatorname{Tr}\left(\mathbf{U}_{\{l,r\}}\mathbf{V}^{*}_{\{l,r\}}{}^{\top}\mathbf{R}_{\{l,r\}}\mathbf{V}^{*}_{\{l,r\}}\mathbf{U}^{\top}_{\{l,r\}}\right) && (2.22)\\
&= \operatorname{Tr}\left(\mathbf{V}^{*}_{\{l,r\}}{}^{\top}\mathbf{R}_{\{l,r\}}\mathbf{V}^{*}_{\{l,r\}}\mathbf{U}^{\top}_{\{l,r\}}\mathbf{U}_{\{l,r\}}\right) && (2.23)\\
&= \operatorname{Tr}\left(\mathbf{V}^{*}_{\{l,r\}}{}^{\top}\mathbf{R}_{\{l,r\}}\mathbf{V}^{*}_{\{l,r\}}\right) && (2.24)\\
&= \operatorname{Tr}\left(\mathbf{\Sigma}^{*}_{\{l,r\}}\right) && (2.25)\\
&= \sum_{i=1}^{Q/2}\lambda^{*}_{\{l,r\},i}\,. && (2.26)
\end{aligned}$$

Clearly the objective is maximized by choosing

$$\begin{cases}\lambda^{*}_{\{l,r\},i} = \lambda_{\{l,r\},i}\\ \mathbf{v}^{*}_{\{l,r\},i} = \mathbf{v}_{\{l,r\},i}\end{cases} \quad \text{for } i = 1, 2, \ldots, Q/2\,. \qquad (2.27)$$

Hence the $Q/2$ optimal spatial filters lie in the subspace spanned by the $Q/2$ generalized eigenvectors corresponding to the $Q/2$ directions with highest power ratio, i.e. with the largest generalized eigenvalues. A straightforward choice for $\mathbf{U}_{\{l,r\}}$ is $\mathbf{U}_{\{l,r\}} = \mathbb{I}$ such that the optimal filters are simply these generalized eigenvectors.

The first direction is then the one with maximal power ratio. The second direction is the one with maximal power ratio but where the output for the attenuated class is uncorrelated to that of the first direction. Subsequent directions similarly look for the maximal power ratio whilst having the output uncorrelated to all previous filters' outputs. Hence the algorithm finds subsequent directions that are in some sense maximally different from one another.

Typically the number of filters per class is taken to be equal for the two classes. Hence with $Q/2$ filters per class, there is a total of Q filters resulting in a Q-dimensional feature vector per window.

2.2.3 Spatio-temporal filtering

Aside from purely spatial filtering for filter extraction, spatial filtering can be combined with temporal filtering (synonymous with spectral filtering) to more selectively extract the relevant features – the SMR's – from each of the channels. Each channel is first filtered in the temporal domain before combining the channels using a spatial filter/pattern. This combined filtering is commonly and most accurately referred to as *spatio-temporal filtering* or *spatio-spectral filtering* although some authors refer to combined temporal and spatial domain filtering simply as spatial filtering.

Common Spatio-Spectral Patterns (CSSP)

To generalize spatial filtering methods, such as CSP analysis, to spatio-temporal filtering, observe that a temporal FIR filter can also be interpreted as a multichannel filter in which each subsequent input channel is a delayed version of the previous channel. Hence in comparison to a purely spatial filter which can be seen as a multichannel filter in which each channel comes from a different location, a spatio-temporal filter has channels that are both from different locations as well as being delayed versions of other channels.

Formally, define $\mathbf{w}_{i,j}$ as the i-th L-tap temporal filter for channel j

$$\mathbf{w}_{i,j} = \begin{bmatrix} w_{i,j,0} & w_{i,j,1} & \dots & w_{i,j,L-1} \end{bmatrix}^\top \in \mathbb{R}^L \,. \tag{2.28}$$

The i-th spatio-temporal filter is then defined by stacking the individual channels' filters:

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{w}_{i,1}^\top & \mathbf{w}_{i,2}^\top & \dots & \mathbf{w}_{i,M}^\top \end{bmatrix}^\top \in \mathbb{R}^{ML} \,. \tag{2.29}$$

Similarly, define $x_j[k]$ to be the sample from channel j at time k. In order to filter with an L-tap (temporal) FIR filter at time k, the samples $x_j[k], x_j[k-1], \dots, x_j[k-L+1]$ are required. Define the stacking of these samples as

$$\mathbf{x}_j[k] = \begin{bmatrix} x_j[k] & x_j[k-1] & \dots & x_j[k-L+1] \end{bmatrix}^\top \tag{2.30}$$

and the stacking of all M channels and their delayed version as

$$\mathbf{x}[k] = \begin{bmatrix} \mathbf{x}_1^\top[k] & \mathbf{x}_2^\top[k] & \dots & \mathbf{x}_M^\top[k] \end{bmatrix}^\top \,. \tag{2.31}$$

The output of the spatial filter $\mathbf{w}_i$ applied to these channels is then

$$\mathbf{y}_i[k] = \sum_{j=1}^{M} \sum_{l=0}^{L-1} w_{i,j,l} \cdot x_j[k-l] = \sum_{j=1}^{M} \mathbf{w}_{i,j}^\top \mathbf{x}_j[k] = \mathbf{w}_i^\top \mathbf{x}[k] \,, \tag{2.32}$$

which is of the same form as spatial filtering. Hence by considering the delayed versions as extra channels, all spatial filtering techniques can straightforwardly be applied in the spatio-temporal domain – albeit with L times as many channels.

Generalizing CSP analysis to spatio-temporal filtering in this way is referred to as *common spatio-spectral pattern (CSSP)* analysis [29]. The covariance matrices now have size $ML \times ML$ instead of $M \times M$ as with purely spatial filtering.

Filterbank CSP (FBCSP)

A disadvantage of this approach is that L needs to be large in order to be sufficiently frequency selective [30]. However the computational complexity is known to increase

rapidly with $\approx L^3$. Furthermore, the increased number of parameters (filter weights) makes the method more prone to overfitting.
Therefore *Filterbank CSP (FBCSP)* analysis first filters each channel into a series of subbands using bandpass filters [30, 31, 32]. The outputs of these filters are then the channels used in the CSP analysis. The resulting optimal filter will then optimally weigh the contribution from each subband w.r.t. other subbands from that channel and w.r.t. other channels' subbands.
This approach allows to make the bandpass filters arbitrarily frequency specific – one could even use IIR filters – without paying for this specificity with a larger parameter space. Rather the size of the parameter space is now the product of the number of channels M and the number of subbands.

Furthermore, because the feature extraction now contains the necessary bandpass filtering for preprocessing (cf. Section 2.1), separate preprocessing filters are no longer necessary. This essentially integrates the preprocessing into the feature extraction, enabling optimal subject-specific pre-processing [31, 32].

Because the two extensions to spatio-temporal filtering discussed above result in a problem of the same type (albeit larger due to the extra channels, which is no different than adding duplicate electrodes), the remainder of this text will not specifically consider the spatio-spectral in the derivations.

2.2.4 Regularization

To avoid overfitting, the objective function of the CSP method[2] can be regularized by adding a penalty on the l_2-norm of the filters [33, 34, 35]. By adding such a penalty term to the denominator of the objective, small norms are favoured when maximizing the objective which is known to reduce overfitting [36]. This technique is commonly known as *Ridge regression* or *Thikhonov regularization.*
The objective function (2.14) is then modified as follows:

$$\mathbf{W}^*_{\{l,r\}} = \arg\max_{\mathbf{W}} \frac{\mathrm{Tr}\left(\mathbf{W}^\top \mathbf{R}_{\{l,r\}} \mathbf{W}\right)}{\mathrm{Tr}\left((1-\lambda)\cdot \mathbf{W}^\top \mathbf{R}_{avg.} \mathbf{W}\right) + \lambda \cdot \sum_{i=1}^{Q/2} \|\mathbf{w}_i\|^2}, \tag{2.33}$$

where $\mathbf{R}_{avg.} = \frac{\mathbf{R}_{\{l,r\}} + \mathbf{R}_{\{r,l\}}}{2}$.

[2]i.e., (2.1) for the MISO case or (2.14) for the MIMO case, as the former is a special case of the latter, the latter will be used for the explanation

Note that this modified problem can also be written as

$$\mathbf{W}^*_{\{l,r\}} = \arg\max_{\mathbf{W}} \frac{\operatorname{Tr}\left(\mathbf{W}^\top \mathbf{R}_{\{l,r\}} \mathbf{W}\right)}{\operatorname{Tr}\left((1-\lambda)\cdot \mathbf{W}^\top \mathbf{R}_{avg.} \mathbf{W}\right) + \lambda \cdot \operatorname{Tr}\left(\mathbf{W}^\top \mathbf{W}\right)} \tag{2.34}$$

$$= \arg\max_{\mathbf{W}} \frac{\operatorname{Tr}\left(\mathbf{W}^\top \mathbf{R}_{\{l,r\}} \mathbf{W}\right)}{\operatorname{Tr}\left(\mathbf{W}^\top \left[(1-\lambda)\, \mathbf{R}_{avg.} + \lambda \cdot \mathbb{I}\right] \mathbf{W}\right)}\,, \tag{2.35}$$

which, similarly to the derivation of Section 2.2.2, is equivalent to calculating the GEVD of $(\mathbf{R}_{\{l,r\}}, (1-\lambda)\,\mathbf{R}_{avg.} + \lambda \cdot \mathbb{I})$:

Observe that whereas the non-regularized objective could be minimized instead to obtain the optimal filter for the other class (cf. supra), the addition of the penalty term prevents prohibits this. Therefore, two GEVD problems will need to be solved to compare the ratios: the GEVD of $(\mathbf{R}_{\{l,r\}}, (1-\lambda)\,\mathbf{R}_{avg.} + \lambda \cdot \mathbb{I})$ and of $(\mathbf{R}_{\{r,l\}}, (1-\lambda)\,\mathbf{R}_{avg.} + \lambda \cdot \mathbb{I})$

Further, note that the use of $\mathbf{R}_{avg.}$ instead of $\mathbf{R}_{\{r,l\}}$ as in (2.14) does not change the solution to the problem but rather makes the parameter λ more interpretable only. This can be seen as follows: the solution of (2.33) is given by the generalized eigenvectors of $\left(\mathbf{R}_{\{l,r\}}, (1-\lambda)\,\mathbf{R}_{avg.} + \lambda \cdot \mathbb{I}\right)$ as was discussed in Section 2.2.2. If $\mathbf{w}$ is such a generalized eigenvector with corresponding eigenvalue σ, then by definition

$$\mathbf{R}_{\{l,r\}}\mathbf{w} = \sigma\left[(1-\lambda)\,\mathbf{R}_{avg.} + \lambda\cdot\mathbb{I}\right]\mathbf{w} = \sigma\left[\frac{1-\lambda}{2}\left(\mathbf{R}_{\{l,r\}} + \mathbf{R}_{\{r,l\}}\right) + \lambda\cdot\mathbb{I}\right]\mathbf{w} \tag{2.36}$$

$$\leftrightarrow \left(1-\sigma\frac{1-\lambda}{2}\right)\mathbf{R}_{\{l,r\}}\mathbf{w} = \sigma\left(\frac{1-\lambda}{2}\mathbf{R}_{\{r,l\}} + \lambda\mathbb{I}\right)\mathbf{w} \tag{2.37}$$

$$\leftrightarrow \mathbf{R}_{\{l,r\}}\mathbf{w} = \frac{2\sigma}{2-\sigma(1-\lambda)}\left(\frac{1-\lambda}{2}\mathbf{R}_{\{r,l\}} + \lambda\mathbb{I}\right)\mathbf{w} = \tag{2.38}$$

$$\leftrightarrow \mathbf{R}_{\{l,r\}}\mathbf{w} = \underbrace{\frac{\sigma(1-\lambda)}{2-\sigma(1-\lambda)}}_{\triangleq\sigma'}\left(\mathbf{R}_{\{r,l\}} + \underbrace{\frac{2\lambda}{1-\lambda}}_{\triangleq\lambda'}\mathbb{I}\right)\mathbf{w} \tag{2.39}$$

$$\leftrightarrow \mathbf{R}_{\{l,r\}}\mathbf{w} = \sigma'\left(\mathbf{R}_{\{r,l\}} + \lambda'\mathbb{I}\right)\mathbf{w}\,. \tag{2.40}$$

It can thus be seen that the objective where $\mathbf{R}_{\{r,l\}}$ is used in the denominator instead of $\mathbf{R}_{avg.}$ yields the same generalized eigenvectors $\mathbf{w}$ but with different but related corresponding eigenvalues σ' and regularization parameter λ'.

2.3 Classification

The final step in the classification pipeline is the classification itself. While non-linear classifiers can use more complex relation between the data to make a prediction of the class, linear classifiers are commonly preferred for MI classification [23, 37]. Their simplicity is often exactly their advantage as simpler models are less prone to overfitting and more robust. They are also amenable to fast implementations and are easy to further regularize.

Therefore, it was opted to use *(Fisher's) Linear Discriminant Analysis (LDA)* which is among the most popular linear classification techniques for MI BCI, next to support vector machines [23].

Similar to CSP, LDA is also formulated as an optimization problem: LDA finds a linear projection direction $\mathbf{a}$ of the data that projects the data to a 1D space[3] that optimally separates the clusters of data corresponding to the classes. Optimal separation is defined as minimal within-class scatter (i.e. the data points are tightly grouped for each class, formally measured by the within-class covariance matrix $\mathbf{\Sigma}_W$) and maximal between-class scatter (i.e. the clusters get mapped to lie far away from each other, formally measured by the between-class covariance matrix $\mathbf{\Sigma}_B$) [36, 38, 39]. [4]

This results in the following optimization problem: Define $\mathbf{m}_l$ and $\mathbf{m}_r$ as the average of the observed features $\mathbf{y}_i$ of resp. class l and class r. Define the within-class covariance matrix $\mathbf{\Sigma}_W$ as

$$\mathbf{\Sigma}_W = \sum_{\mathbf{y}_i \in l} (\mathbf{y}_i - \mathbf{m}_l)(\mathbf{y}_i - \mathbf{m}_l)^\top + \sum_{\mathbf{y}_i \in r} (\mathbf{y}_i - \mathbf{m}_r)(\mathbf{y}_i - \mathbf{m}_r)^\top \tag{2.41}$$

and the between-class covariance matrix $\mathbf{\Sigma}_B$ as

$$\mathbf{\Sigma}_B = (\mathbf{m}_l - \mathbf{m}_r)(\mathbf{m}_l - \mathbf{m}_r)^\top . \tag{2.42}$$

Take the solution as the linear projection $z = \mathbf{a}^\top \mathbf{y}$ that maximizes

$$\max_{\mathbf{a}} \frac{\mathbf{a}^\top \mathbf{\Sigma}_W \mathbf{a}}{\mathbf{a}^\top \mathbf{\Sigma}_B \mathbf{a}} , \tag{2.43}$$

which can be seen to be an optimization problem of the same form as the CSP objective (see (2.1)). Consequently it is similarly solved by taking $\mathbf{a}$ as the generalized eigenvector corresponding to the largest generalized eigenvalue, as was shown for CSP.

By next setting a threshold z_0, for example based on the class conditional densities, this yields a classifier.

Similar to CSP, this objective can also be regularized to avoid overfitting on the training data. As the classification is not the main focus of this thesis however, the reader is referred to [36, 39] for more details.

[4]LDA can be extended to higher dimensional projections similar to how CSP was extended to multiple filters, yielding a method for dimensionality reduction.

Chapter 3

Distributed feature extraction

As discussed in the introduction of this text, the typical MI classification system as presented in the previous chapter is rather ill-suited for long term use. Therefore, this text evaluates the use of a *wireless EEG sensor network (WESN)* as a more suitable alternative for this use-case to conventional EEG acquisition. To make this approach feasible however, the classification pipeline of the previous chapter will have to be adapted using distributed signal processing techniques. Therefore, this adapted pipeline that enables efficient (w.r.t. the goals set out in Section 1.4) MI classification on a WESN will sometimes be referred to as the distributed (classification) pipeline in this text.

This chapter presents the architecture of the distributed MI classification pipeline to efficiently run on a WESN as envisioned. The central pipeline of the previous chapter will be used as a starting point and framework for this distributed version.

In the first section of this chapter, more details will first be given on the WESN as envisioned in this text. This section also includes an overview of the different network topologies considered as well as the standard notation concerning WESN. In the next section, a naive approach to running the complete MI classification pipeline on a WESN (i.e. without distributed signal processing) will be discussed that serves as a baseline for the more efficient approaches further on in the chapter.

The following three sections are then concerned with efficient distributed implementations of the feature extraction step of the classification pipeline.
The first of the three sections details the derivations for both considered topologies for CSP without regularization nor re-use of the data (cf. infra). The next section then extends the algorithms to include regularization. Finally the last of these three section adds the efficient re-use of data to efficiently be able to increase the tracking speed of the algorithm.

The final section of this chapter discusses how the complete classification pipeline can then be setup efficiently in a distributed fashion. It will turn out that this only requires a modification to the feature extraction step for which then the algorithms of the three previous sections will be used.

Remark: To be as general as possible without needlessly complicating the notation, the feature extraction algorithms to be described in this chapter are presented with multiple filters per class and eventually the addition of Thikhonov regularization (see respectively Sections 2.2.2 and 2.2.4), but without the extension to spatio-temporal filtering as this does not alter the algorithms in any way except for the dimensions of the filters and covariance matrices (see Section 2.2.3).

3.1 Wireless sensor networks

Before considering modifications to the central architecture of the previous chapter, this section first defines the envisioned network in more detail and presents the notation concerning such a network that will then be used throughout the remainder of this text.

The system envisaged in this thesis uses a wireless EEG sensor network (WESN) consisting of multiple low-power sensor nodes attached to the head of the wearer. Each node is battery-powered and consists of an electrode array, a small DSP processor and a wireless network stack used to form a network with the other nodes or with a fusion centre (FC). Since every node is equipped with a processor, the system is amenable to distributed processing which could help with per-node requirements.

The communication between the nodes requires the implementation of a network protocol stack with protocols suited for the characteristics of the WESN. The media layers of the protocol stack provide a.o. protocols for energy efficient radio transmission (physical layer[1]), packet handling and addressing (data link layer), maintaining the desired network topology even in case of the removal or addition of nodes (network layer) and reliability (transport layer). These are however outside of the scope of this text.
The algorithms which are discussed in this text are situated in the application layer (or presentation layer if used, but more generally in one of the the host layers). Owing to the layer model, the media layers can be abstracted and considered to provide a reliable wireless network in which each node can readily receive and transmit high level data to its neighbours in the desired topology. However by removing some of the abstraction, link failures or creations can be handled more efficiently [40]. This is however outside of the scope of this text.

While this text focusses on extracutaneous nodes and electrodes, a similar sensor

[1]As defined by the OSI network model

network would for example also be useful for subdermal nodes which would reduce (skin) movement artifacts, guarantee correct electrode positioning and reduce the amount of visible equipment albeit at the cost of a.o. invasiveness and more difficult power delivery. Potentially even intracranial electrodes, i.e. electrocorticography (ECoG), could benefit from a sensor network to connect multiple electrode grids to one-another to do the processing. Compared to EEG, the spatial and spectral resolution of ECoG electrodes is significantly higher allowing for more powerful methods concerning small areas of the brain. The main disadvantages of ECoG are its invasiveness [41] – usually requiring a craniotomy – and biological issues affecting the signal quality in the long term [42, 43].

Notation

The set of nodes in the WESN will be denoted to as $\mathcal{K}$. The size K of the network is defined as the number of nodes in the network and follows from the cardinality of $\mathcal{K}$, i.e. $K = |\mathcal{K}|$. The set of neighbours of a node k is denoted as $\mathcal{N}_k$ and is defined as the set of node to which node k has a direct communication link.
A node $k \in \mathcal{K}$ has M_k electrodes of which the observations at time j will be referred to with the vector $\mathbf{x}_k[i] \in \mathbb{R}^{M_k}$. The total amount of electrodes in the network will be denoted as $M = \sum_{k=1}^{K} M_k$. The observations from all nodes are stacked in the vector $\mathbf{x}[i] \in \mathbb{R}^M$.

With respect to the two classes, define $\mathbf{x}_{k,\{l,r\}}[i]$ as an observation $\mathbf{x}_k$ that belongs to class $\{l, r\}$ where the brace-notation of the previous chapter has been used to denote the possible options for the subscript.

The index i will often be dropped for notational brevity.

Network topology

The network formed by the sensor nodes can form different topologies based on how the nodes interconnect. In this work, two topologies are considered: fully connected and tree topologies.

In a fully connected topology, each nodes has a direct (wireless) link with every other node in the network. As will be shown later, these direct connections allow data to be shared with the other nodes quickly. Figure 3.1a gives a schematic example of a fully connected topology.

The second topology considered is the tree topology as schematically illustrated in Figure 3.1b. A tree topology has two main advantages as compared to a fully connected topology. First, depending on the construction of the tree, link distances can be kept short. Since, as will be explained in Section 4.4, transmission power P relates to the distance d as $P \propto d^n$ with $n = 5$-6 this could mean a much lower energy consumption. Second, as compared to other conceivable topologies, there are no loops. This makes it easy to distribute data in the network as loops can cause convergence problems for the distributed algorithms to be described later in this chapter [14].

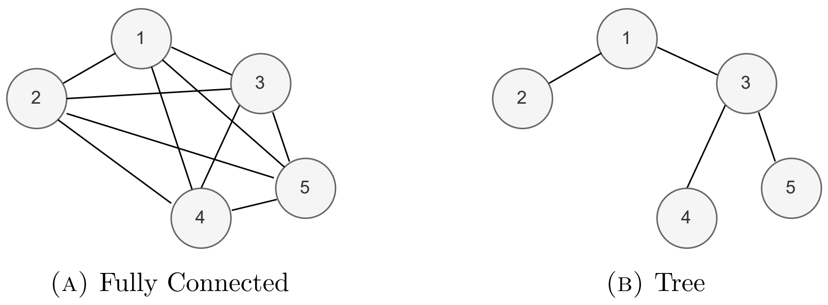


FIGURE 3.1: Example of a WESN organized as either a fully connected (a) or tree topology (b). The lines between the nodes, represented as circles, indicate the internode connections.

The disadvantage is that data may need to traverse multiple in order to reach another node in the network. The straightforward approach of having nodes relay data not destined for them to nodes that connect to the destination however, is not scalable as it requires extra energy, especially from nodes close to the root of the tree, to do the relaying as well as increasing the latency from source to destination [44]. Thus, the algorithm that operates on a tree topology that will be described later in this chapter, will use a cooperative approach instead that lets a node incorporate data it has been sent into its own.
Furthermore, because there is only a single path between any two nodes, a tree topology is less resilient against link failures than other topologies

It should be noted that any topology can be pruned into a tree topology using for example Dijkstra's shortest path algorithm [45]. Therefore algorithms that work on a tree topology can operate on all possible topologies after pruning. Hence such algorithms are sometimes referred to as *topology independent (TI)* in the literature [46].

Dijkstra's shortest path algorithm guarantees that the distance to each node starting from the root node is minimal, where the distance is more accurately described as a cost which can be defined by any cost function. In the original algorithm, this was taken to be the physical distance, but later in this text, the communication related energy of each link will be used as a more relevant cost.
As this optimality is only guaranteed for paths starting from a root node (and paths that so happen to be included in that tree due to Bellman's principle of optimality), a different pruned tree should be used depending on the destination of the transmissions to be optimal. This approach will be referred to as a *dynamic topology.*

Although this approach is more efficient since it can exploit every possible connection between nodes, it also increases the amount of overhead of the network layer. Furthermore, as will be discussed in Section 3.5, the fact that the neighbours of the nodes are continuously changing also eliminates properties the algorithm can use. Therefore, another approach would be to prune the network down to a single tree regardless of source/destination path optimality. This approach will be referred to as a *static topology.*

3.2 Centralized algorithm on a WESN

A naive approach to MI classification within a WESN would be to centralize all signals in one of the nodes, called the *fusion center (FCe)* or symbolically k_{FCe}, and run the central classification pipeline (as described in the previous chapter) on that node. This approach will be the baseline for comparison with more efficient approaches.

Algorithm 3.1 details all steps of the classification pipeline as presented in the previous chapter, for this naive implementation. For notational clarity, we introduce the notation $\mathbf{x}^i[j] = \mathbf{x}[iN + j]$ to denote the j-th sample from the i-th window.
In each window, observations from only one of the classes is available. Updating the covariance matrices is hence done in the same way as discussed in Section 2.2 the previous chapter.

Algorithm 3.1: Sliding-window adaptive CSP

```
Set $i \leftarrow 0$
repeat
    Collect a window of $N$ observations from all $M$ electrodes:
      $\mathbf{x}^i[j] = \mathbf{x}[iN + j] \in \mathbb{R}^M$ where $j = 1, \dots, N$
    Each node $k \in \mathcal{K} \setminus \{k_{FCe}\}$ transmits its observations to node $k_{FCe}$
    Filter signals: $\bar{\mathbf{x}}^i[j] = {\mathbf{W}^i}^\top \mathbf{x}^i[j]$
    Calculate features as $\mathbf{y}^i = \log \left( \frac{1}{N} \sum_{j=1}^{N} \bar{\mathbf{x}}^i[j]^2 \right)$
    Predict the output class
    Update the covariance matrices (see section 2.2), corresponding to the
      correct label 'l' or 'r'
    Update the filters by solving the GEVD of the covariance matrices to
      obtain the $Q/2$ principal eigenvectors of $(\mathbf{R}_l^i + \alpha \cdot \mathbb{I}, \mathbf{R}_r^i)$ and of
      $(\mathbf{R}_r^i + \alpha \cdot \mathbb{I}, \mathbf{R}_l^i)$ to form $\mathbf{W}^{i+1} \in \mathbb{R}^{Q \times M}$
    Resolve sign ambiguity;
    $i \leftarrow i + 1$
end
```

3.3 Distributed Common Spatial Patterns

Central to the CSP method used for feature extraction in the classification pipeline, is the GEVD of the network-wide covariance matrices corresponding to the two classes. However as shown in [14] and as will be discussed in Chapter 5, the centralization of the signals in a single node to be able to compute those covariance matrices is prohibitively expensive in terms of energy usage.

To combat the prohibitively large communication-related energy consumed by centralizing the observations, Bertrand and Moonen propose an iterative method called *Distributed Adaptive Covariance-matrix Generalized Eigenvector Estimation (DACGEE)* that finds the generalized eigenvectors of a pair of covariance matrix via an iterative procedure over consecutive windows of observations.

Since CSP boils down to a GEVD, in this context the algorithm optimizes the (MIMO) CSP objective in (2.15) by iteratively updating a filter matrix $\mathbf{W}^i$ without requiring centralization of the observations to compute the network-wide covariance matrices [20].

In [20] the authors prove that for $i \to \infty$, the centralized optimum $\mathbf{W}^*$ as computed by the methods of Section 2.2 is reached. More formally,

$$\mathbf{W}^* = \lim_{i\to\infty} \mathbf{W}^i \, .$$

In the paper the authors derive the method for the case of a fully connected network topology. However, based on similar previous work, they also propose a generalization towards tree topologies. The derivations for respectively fully connected and for the generalization to tree topologies will be discussed in the next two subsections.

3.3.1 Fully connected topologies

The main idea behind the algorithm is that in each iteration/window, all nodes in the network compress their observations of that window using the part of the filter matrix iterate corresponding to the node. Next, these compressed observations are then broadcasted to the rest of the network, rather than the raw observations as in the naive algorithm (cf. supra). In each iteration, one of the nodes then computes the compressed covariance matrix and updates the filter matrix. Therefore, this last node will be referred to as the *updating node*, usually symbolically denoted as node q.

Derivation

The following derivation is taken from the original paper with slight notational adaptations to be consistent with the rest of this text as well as some changes in the ordering of the steps of the derivation.

First consider the following partitioning of $\mathbf{W}$[2]:

$$\mathbf{W} = \left[\begin{array}{cccc} \mathbf{W}_1^\top & \mathbf{W}_2^\top & \dots & \mathbf{W}_K^\top \end{array}\right]^\top , \tag{3.1}$$

where $\mathbf{W}_k \in \mathbb{R}^{M_k \times Q/2}$. This partitions the filter matrix into smaller filter submatrices $\mathbf{W}_k$ corresponding to the M_k channels of node $k \in \mathcal{K}$.

Recall the constrained MIMO objective from eq. (2.15) repeated here for convenience:

$$\begin{aligned} \mathbf{W}^*_{\{l,r\}} = \arg\max_{\mathbf{W}} \ &\mathrm{Tr}\left(\mathbf{W}^\top \mathbf{R}_{\{l,r\}} \mathbf{W}\right) \\ \text{s.t.} \quad &\mathbf{W}^\top \mathbf{R}_{\{r,l\}} \mathbf{W} = \mathbb{I}\,. \end{aligned}$$

This optimization problem is then transformed into an iterative procedure using an alternate optimization (AO) procedure:

1. Set $i \leftarrow 0,\ q \leftarrow 1$ and $\mathbf{W}^0_{\{l,r\}}$ as a random $M \times Q/2$ matrix
2. Solve
$$\begin{aligned} \mathbf{W}^{i+1}_{\{l,r\}} = &\arg\max_{\mathbf{W}} \ \mathrm{Tr}\left(\mathbf{W}^\top \mathbf{R}_{\{l,r\}} \mathbf{W}\right) \\ \text{s.t.} \quad &\begin{array}{c} \mathbf{W}^\top \mathbf{R}_{\{r,l\}} \mathbf{W} = \mathbb{I} \\ \mathcal{R}(\mathbf{W}_k) = \mathcal{R}(\mathbf{W}^i_{k,\{l,r\}}) \quad \forall k \in \mathcal{K} \setminus \{q\} \end{array} \end{aligned} \tag{3.2}$$
where $\mathbf{W}_k$ is the k-th submatrix of $\mathbf{W}$ similarly defined as the partition given in eq. (3.1) and $\mathcal{R}(\cdot)$ denotes the range of its argument.
3. Set $i \leftarrow i + 1$ and $q \leftarrow (q \ mod \ K) + 1$
4. Return to step 2.

In each iteration of the AO procedure, $\mathbf{W}_q$ is optimized subject to only the original constraint while the other submatrices $\mathbf{W}_k$ are also constrained to have the same range.
Note that thus far, the AO procedure still requires centralizing all observations to compute the covariance matrices.

The second constraint of the optimization problem in equation 3.2 can equivalently written in terms of the matrices $\left\{\mathbf{G}_k \in \mathbb{R}^{Q/2 \times Q/2} | \forall k \in \mathcal{K} \setminus \{q\}\right\}$ as

$$\mathbf{W}_k = \mathbf{W}^i_{k,\{l,r\}} \mathbf{G}_k\,, \tag{3.3}$$

[2]Recall from Section 2.2.2 that we have defined Q as the total amount of filters and hence there are $Q/2$ per class

allowing to reparametrize the optimization variable as

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^i_{1,\{l,r\}}\mathbf{G}_1 \\ \vdots \\ \mathbf{W}^i_{q-1,\{l,r\}}\mathbf{G}_{q-1} \\ \mathbf{W}_k \\ \mathbf{W}^i_{q+1,\{l,r\}}\mathbf{G}_{q+1} \\ \vdots \\ \mathbf{W}^i_{K,\{l,r\}}\mathbf{G}_K \end{bmatrix} . \tag{3.4}$$

Now define the stacking of all optimization variables

$$\widetilde{\mathbf{W}}_q = \begin{bmatrix} \mathbf{W}_q^\top & \mathbf{G}_1^\top & \dots & \mathbf{G}_{q-1}^\top & \mathbf{G}_{q+1}^\top & \dots & \mathbf{G}_K^\top \end{bmatrix}^\top , \tag{3.5}$$

and define the following matrix

$$\mathbf{C}^i_{k,\{l,r\}} = \begin{bmatrix} \mathbf{0} & \mathbf{B}^i_{<k,\{l,r\}} & \mathbf{0} \\ \mathbf{I}_{M_k} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}^i_{>k,\{l,r\}} \end{bmatrix} , \tag{3.6}$$

where

$$\mathbf{B}^i_{<k,\{l,r\}} = \text{blkdiag}\left(\mathbf{W}^i_{1,\{l,r\}}, \dots, \mathbf{W}^i_{k-1,\{l,r\}}\right) , \tag{3.7}$$

$$\mathbf{B}^i_{>k,\{l,r\}} = \text{blkdiag}\left(\mathbf{W}^i_{k+1,\{l,r\}}, \dots, \mathbf{W}^i_{K,\{l,r\}}\right) , \tag{3.8}$$

such that (3.4) can be compactly written as

$$\mathbf{W} = \mathbf{C}^i_{q,\{l,r\}}\widetilde{\mathbf{W}}_q . \tag{3.9}$$

Returning to the optimization problem in (3.2), the second constraint can now be eliminated by reparametrizing the problem i.f.o. $\widetilde{\mathbf{W}}_q$ due to (3.3-3.5). For notational clarity, define $\mathbf{x}^i_{k,\{l,r\}}[j] = \mathbf{x}^i_{k,\{l,r\}}[iN + j]$ to denote the j-th sample of the window corresponding to iteration i and similarly define $\mathbf{x}^i_{\{l,r\}}$ as the stacking of such vectors. The index j will often be dropped for notational brevity.

The objective is then rewritten as

$$\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \underset{\widetilde{\mathbf{W}}_q}{\arg\max}\ \text{Tr}\left(\widetilde{\mathbf{W}}_q^\top {\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{R}_{\{l,r\}}\mathbf{C}^i_{q,\{l,r\}}\widetilde{\mathbf{W}}_q\right) , \tag{3.10}$$

where

$${\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{R}_{\{l,r\}}\mathbf{C}^i_{q,\{l,r\}} = E\left[\left({\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{x}^i_{\{l,r\}}\right)\left({\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{x}^i_{\{l,r\}}\right)^\top\right] \tag{3.11}$$

with

$$\mathbf{C}_{q,\{l,r\}}^{i}{}^{\top}\mathbf{x}_{\{l,r\}}^{i} = \begin{bmatrix} \mathbf{x}_{q,\{l,r\}}^{i} \\ \mathbf{W}_{1,\{l,r\}}^{i}{}^{\top}\mathbf{x}_{1,\{l,r\}}^{i} \\ \vdots \\ \mathbf{W}_{q-1,\{l,r\}}^{i}{}^{\top}\mathbf{x}_{q-1,\{l,r\}}^{i} \\ \mathbf{W}_{q+1,\{l,r\}}^{i}{}^{\top}\mathbf{x}_{q+1,\{l,r\}}^{i} \\ \vdots \\ \mathbf{W}_{K,\{l,r\}}^{i}{}^{\top}\mathbf{x}_{K,\{l,r\}}^{i} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{q,\{l,r\}}^{i} \\ \bar{\mathbf{x}}_{1,\{l,r\}}^{i} \\ \vdots \\ \bar{\mathbf{x}}_{q-1,\{l,r\}}^{i} \\ \bar{\mathbf{x}}_{q+1,\{l,r\}}^{i} \\ \vdots \\ \bar{\mathbf{x}}_{K,\{l,r\}}^{i} \end{bmatrix}, \tag{3.12}$$

where $\bar{\mathbf{x}}_{k,\{l,r\}}^{i} = \mathbf{W}_{k,\{l,r\}}^{i}{}^{\top}\mathbf{x}_{k,\{l,r\}}^{i}$ are the observations of the channels local to node k, filtered with the corresponding part $\mathbf{W}_{k,\{l,r\}}^{i}$ of the filter in iteration i. Further define the following notation:

$$\bar{\mathbf{x}}_{-k,\{l,r\}}^{i} = \begin{bmatrix} \bar{\mathbf{x}}_{1,\{l,r\}}^{i} \\ \vdots \\ \bar{\mathbf{x}}_{k-1,\{l,r\}}^{i} \\ \bar{\mathbf{x}}_{k+1,\{l,r\}}^{i} \\ \vdots \\ \bar{\mathbf{x}}_{K,\{l,r\}}^{i} \end{bmatrix} \text{and} \tag{3.13}$$

$$\widetilde{\mathbf{x}}_{k,\{l,r\}}^{i} = \begin{bmatrix} \mathbf{x}_{k,\{l,r\}} \\ \bar{\mathbf{x}}_{-k,\{l,r\}}^{i} \end{bmatrix}. \tag{3.14}$$

$\bar{\mathbf{x}}_{-k,\{l,r\}}^{i}$ is then the stacking of all compressed signals received by node k in iteration i. Hence it is of length $(K-1)Q/2$. $\widetilde{\mathbf{x}}_{k,\{l,r\}}^{i}$ stacks node k's own M_k signals on top to concatenate all signals available to node k in a vector of length $M_k + (K-1)Q/2$. For notational brevity later on in this text, the total number of available channels (both from its own electrodes as well as compressed channels from others') will be denoted as $\widetilde{M_k} \triangleq M_k + (K-1)Q/2$, such that $\widetilde{\mathbf{x}}_{k,\{l,r\}}^{i} \in \mathbb{R}^{\widetilde{M}_k}$.

Using these definitions, it is clear that

$$\mathbf{C}_{q,\{l,r\}}^{i}{}^{\top}\mathbf{x}_{\{l,r\}} = \widetilde{\mathbf{x}}_{q,\{l,r\}}^{i} \tag{3.15}$$

and hence

$$\mathbf{C}_{q,\{l,r\}}^{i}{}^{\top}\mathbf{R}_{\{l,r\}}\mathbf{C}_{q,\{l,r\}}^{i} = E\left[\widetilde{\mathbf{x}}_{q,\{l,r\}}^{i}(\widetilde{\mathbf{x}}_{q,\{l,r\}}^{i})^{\top}\right] \triangleq \widetilde{\mathbf{R}}_{q,\{l,r\}}^{i}. \tag{3.16}$$

The objective is then compactly written as

$$\mathrm{Tr}\left(\widetilde{\mathbf{W}}_{q}^{\top}\widetilde{\mathbf{R}}_{q,\{l,r\}}^{i}\widetilde{\mathbf{W}}_{q}\right). \tag{3.17}$$

Similarly, the constraint $\mathbf{W}^\top \mathbf{R}_{\{r,l\}}\mathbf{W} = \mathbb{I}$ can be rewritten i.f.o. $\widetilde{\mathbf{W}}_q$ as

$$\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}^i_{q,\{r,l\}} \widetilde{\mathbf{W}}_q = \mathbb{I}\,, \tag{3.18}$$

where $\widetilde{\mathbf{R}}^i_{q,\{r,l\}}$ is similarly defined as in (3.16).

Hence the optimization problem (3.2) in each iteration is equivalently solved by finding $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ as the solution of the following compressed optimization problem:

$$\begin{aligned} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \underset{\widetilde{\mathbf{W}}_q}{\arg\max}\ & \operatorname{Tr}\left(\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}_q\right) \\ \text{s.t.}\quad & \widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}^i_{q,\{r,l\}} \widetilde{\mathbf{W}}_q = \mathbb{I} \end{aligned} \tag{3.19}$$

and setting $\mathbf{W}^{i+1}_{\{l,r\}} = \mathbf{C}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$. Note that this compressed problem has the same form as the original problem (2.15) which was proven to be equivalent to a GEVD problem (see section 2.2.2). Therefore, the compressed problem can be solved by performing a GEVD of $\left(\widetilde{\mathbf{R}}^i_{q,\{l,r\}}, \widetilde{\mathbf{R}}^i_{q,\{r,l\}}\right)$.

To avoid the sign of the filters flipping between iterations – as the solution to the GEVD is only unique up to the sign (the length was already fixed by adding a constraint to the objective, as first discussed in Section 2.2.1) – the filter matrix $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ is taken such that the signs of the compressed filter matrix corresponding to node q match between iterations i and $i+1$ [46]. This is achieved by

$$\begin{aligned} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} &= \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}} \mathbf{D}^* \\ \text{where}\quad \mathbf{D}^* &= \underset{\mathbf{D}\in\mathcal{D}}{\arg\min} \left\| \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} \mathbf{D} - \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}} \right\|_F \end{aligned} \tag{3.20}$$

where $\mathcal{D}$ represents the class of $Q/2 \times Q/2$ signature matrices, i.e. the class of $Q/2 \times Q/2$ diagonal matrices for which all diagonal elements are elements of $\{-1, 1\}$.

The covariance matrices in this compressed problem are those of the vector $\widetilde{\mathbf{x}}^i_{q,\{l,r\}}$ which is comprised of observations from the channels local to node q and of compressed observations of the other nodes. Hence all observations need not be centralized in node q as they would need to be with the naive centralized algorithm to estimate the network-wide covariance matrix. Rather each node compresses its observations and sends only those to the updating node q. Estimation of the compressed covariance matrices can be done similar to the central case:

$$\widetilde{\mathbf{R}}^i_{q,\{l,r\}} \approx \frac{1}{N_{\{l,r\}}} \sum \widetilde{\mathbf{x}}^i_{q,\{l,r\}} (\widetilde{\mathbf{x}}^i_{q,\{l,r\}})^\top\,, \tag{3.21}$$

with $N_{\{l,r\}}$ the amount of samples corresponding to class $\{l, r\}$ in the current window. Indeed, as the observations are compressed using $\mathbf{W}^i$ which changes in each iteration, the covariance matrices will need to be estimated in each iteration as well. Note that this requires a window of new observations from both classes in each iteration.

By parametrizing $\mathbf{W}$ such that the filters of the channels local to q can be freely chosen and that the filters of other nodes' signals can only be transformed within the same range using transformations $\mathbf{G}_k$, the compressed signals of the latter set of nodes are all that is required for one iteration. Indeed, the transformation $\mathbf{G}_k$ can equivalently be applied to the compressed signals instead of to the filter matrix $\mathbf{W}_k$ first.
For $Q < M_k$, this method reduces the communication-related energy usage of node k w.r.t. the naive approach represented in the previous section, but requires multiple iterations in order to converge towards the central optimum $\mathbf{W}^*$.
Furthermore, as new observations are used to calculate the (compressed) covariance matrices in each iteration, the algorithm can also track slowly varying statistics – it is an adaptive algorithm. However if this change is too fast w.r.t. the convergence speed, the algorithm never converges to the optimum for the statistics at that point in time [47].
In Section 3.5 the method of this section will be extended to efficiently run multiple iterations on the same set of observations which will introduce a tunable trade-off between tracking performance and communication-related energy consumption.

Note that if the nodes broadcast their compressed observations on the network rather than solely transmitting them to node q and if node q also broadcasts its compressed observations on the network, every node is able to construct the filtered output and hence predict the class.

Putting all of these steps together results in the DACGEE algorithm as given in Alg. 3.2.

3.3.2 Tree topologies

In the same DACGEE paper [20], the authors also mention the possibility of extending the method to partially connected networks, such as tree networks, and refer to [48] were a method was proposed for estimating the covariance matrix eigenvector in a distributed way in both fully connected and partially connected topologies. DACGEE can be seen as a generalization of the method in that work and hence the procedure for extending the algorithm to partially connected network is indeed analogous to the one described in that paper. This will be discussed in this subsection.

Data flow

As discussed in Section 3.1 a node in a tree network only communicates with its neighbours. Consequently, the approach taken with fully connected networks in which each node broadcasts the compressed observations is not applicable.

Instead, a node k sends a neighbouring node $n \in \mathcal{N}_k$, the sum of its own compressed signals and those received from its neighbours except for ln (to avoid feedback loops, see [48] for a detailed explanation) for both classes. Let $\mathbf{z}^i_{k\to n,\{r,l\}}$ denote the signal

Algorithm 3.2: DACGEE [20]

```
Set $i \leftarrow 0,\ q \leftarrow 1$ and initialize all $\mathbf{W}^0_{k,\{l,r\}}\ \forall k \in \mathcal{K}$ as random matrices
repeat
    Each node $k \in \mathcal{K}$ compresses and broadcasts $N$ new observations
    $\bar{\mathbf{x}}^i_{k,\{l,r\}} = {\mathbf{W}^i_{k,\{l,r\}}}^\top \mathbf{x}^i_{k,\{l,r\}}$ and $\bar{\mathbf{x}}^i_{k,\{r,l\}} = {\mathbf{W}^i_{k,\{r,l\}}}^\top \mathbf{x}^i_{k,\{r,l\}}$
    At node $q$:
      • Estimate $\widetilde{\mathbf{R}}_{q,\{l,r\}}$ and $\widetilde{\mathbf{R}}_{q,\{r,l\}}$
      • Solve
            $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \arg\max_{\widetilde{\mathbf{W}}} \operatorname{Tr}\left(\widetilde{\mathbf{W}}^\top \widetilde{\mathbf{R}}_{q,\{l,r\}} \widetilde{\mathbf{W}}\right)$
            s.t. $\widetilde{\mathbf{W}}^\top \widetilde{\mathbf{R}}_{q,\{r,l\}} \widetilde{\mathbf{W}} = \mathbb{I}$
      • Resolve the sign ambiguity
            $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \widetilde{\mathbf{W}}^i_{q,\{l,r\}} \mathbf{D}^*$
            where $\mathbf{D}^* = \arg\min_{\mathbf{D} \in \mathcal{D}} \left\| \mathbf{W}^{i+1}_{q,\{l,r\}} \mathbf{D} - \mathbf{W}^i_{q,\{l,r\}} \right\|_F$
      • Define $P = (K-1)\frac{Q}{2}$ and partition $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ as
            $\mathbf{W}^{i+1}_{q,\{l,r\}} = \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$
            $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{0} & \mathbb{I}_P \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$
      • Broadcast $\mathbf{G}_{-q}$ to all other nodes
    Each node $k \in \mathcal{K} \setminus \{q\}$ updates
            $\mathbf{W}^{i+1}_{k,\{l,r\}} = \mathbf{W}^i_{k,\{l,r\}} \mathbf{G}_k$
      where $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{G}_1^\top & \dots & \mathbf{G}_{q-1}^\top & \mathbf{G}_{q+1}^\top & \dots & \mathbf{G}_K^\top \end{bmatrix}^\top$
    Set $i \leftarrow i+1$ and $q \leftarrow (q\ mod\ K) + 1$
end
```

transmitted by node k to node $n \in \mathcal{N}_k$ in iteration i for class $\{r, l\}$. This signal is then defined as

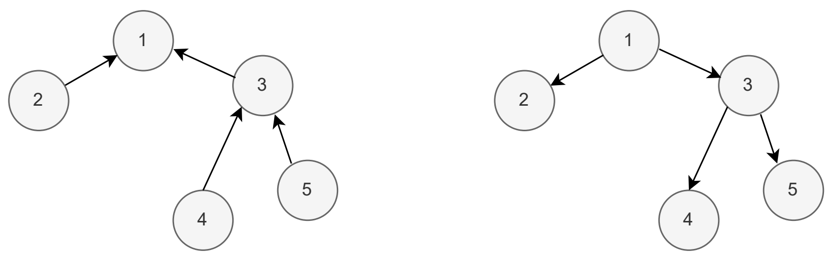
$$\mathbf{z}^i_{k\to n,\{l,r\}} = {\mathbf{W}^i_{k,\{l,r\}}}^\top \mathbf{x}^i_{k,\{l,r\}} + \sum_{m \in \mathcal{N}_k \setminus \{n\}} \mathbf{z}^i_{m\to k,\{l,r\}}\,. \tag{3.22}$$

Note that this definition gives rise to a fusion and a diffusion stage (to and from the root node) in the transmission of the data.
Indeed, the leaf nodes (i.e. the nodes k for which $|\mathcal{N}_k| = 1$) start the flow by transmitting their compressed local observations to their sole neighbour. Those nodes combine the transmissions from their leaf node children and transmit to the

node that connects them with the root. This continues in every layer, each time combining the data further up the tree towards the root of the tree. The data is said to fuse towards the root, hence this stage is called the fusion stage.
Once the data has reached the root on each of its neighbours, the root node can combine and transmit the correct signal back down to its neighbours towards the leaves of the tree. A downstream node receives this transmission and diffuses the correctly combined signal to each of its neighbours except for the upstream node. The data diffuses to the leaves and hence this stage is called the diffusion stage.

Similar to the fully connected case, by completing this flow (which does more transmissions than is strictly necessary for the algorithm that follows) each node has all compressed signals and hence every node is once again able to construct the filtered output from these compressed observations only.

Figure 3.2 shows a schematic illustration of the two stages for the example network of figure 3.1b.



(A) Fusion flow (B) Diffusion flow

FIGURE 3.2: Illustration of the natural data flow in a tree network. The tree network is the same network as in figure 3.1b.

Now define the vector $\widetilde{\mathbf{x}}^i_{-k,\{l,r\}}$ (compare to (3.14)) which stacks all signals that node k has available to it as

$$\widetilde{\mathbf{x}}^i_{-k,\{l,r\}} = \begin{bmatrix} \mathbf{x}_{k,\{l,r\}} \\ \bar{\mathbf{z}}^i_{\to k,\{l,r\}} \end{bmatrix} , \tag{3.23}$$

where $\bar{\mathbf{z}}^i_{\to k,\{l,r\}}$ is defined as the stacking of the signals $\{\bar{\mathbf{z}}^i_{l\to k,\{l,r\}} | l \in \mathcal{N}_k\}$. The latter vector is now of length $\|\mathcal{N}_k\|Q/2$ while the former of length $M_k + \|\mathcal{N}_k\|Q/2$ which again will be denoted to as $\widetilde{M}_k$ as in Section 3.3.1, generalizing the definition.

Next define $\mathcal{V}_{ko}$ as the set of nodes that gets disconnected from the network if the connection between nodes k and o is cut. Observe that then

$$\mathbf{z}^i_{k\to o,\{l,r\}} = {\mathbf{W}^i_{k,\{l,r\}}}^\top \mathbf{x}^i_{\{l,r\}} + \sum_{m\in\mathcal{V}_{ko}} {\mathbf{W}^i_{m,\{l,r\}}}^\top \mathbf{x}^i_m . \tag{3.24}$$

Hence in $\bar{\mathbf{x}}^i_{\to k,\{l,r\}}$ all observations are available either uncompressed if they are local to node k or combined and compressed if they have been received via a neighbouring node.

Derivation

As a generalization of DACGEE, the derivation of the extension of the method to tree topologies is based on transforming the CSP objective into the same alternate optimization procedure of the previous section. This extension of the algorithm will further be denoted as t-DACGEE.

First, define a bijection f_q that pairs each node $n_o \in \mathcal{N}_q$ with a number $o \in \{1, \ldots, |\mathcal{N}_q|\}$. More formally:

$$f_q : \mathcal{N}_q \to \{1, \ldots, |\mathcal{N}_q|\} : n_o \to o \,. \tag{3.25}$$

For the updating node q, the definition of $\widetilde{\mathbf{x}}^i_{\to q,\{l,r\}}$ induces a parametrization similar to the previous section of the filter vector $\mathbf{W}$ of the CSP objective such that the optimization problem of the AO procedure is equivalently written in terms of (compare to (3.5)):

$$\widetilde{\mathbf{W}}_q = \begin{bmatrix} \mathbf{W}_q^\top & \mathbf{G}_1^\top & \ldots & \mathbf{G}_{|\mathcal{N}_q|}^\top \end{bmatrix}^\top \,. \tag{3.26}$$

$\mathbf{W}$ then relates to $\widetilde{\mathbf{W}}_q$ as

$$\mathbf{W} = \mathbf{C}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}_q \,, \tag{3.27}$$

where (compare to eq. (3.6))

$$\mathbf{C}^i_{q,\{l,r\}} = \begin{bmatrix} \mathbf{0} & \mathbf{B}^i_{<q,\{l,r\}} \\ \mathbf{I}_{M_q} & \mathbf{0}_{M_q \times Q/2|\mathcal{N}_q|} \\ \mathbf{0} & \mathbf{B}^i_{>q,\{l,r\}} \end{bmatrix} \tag{3.28}$$

The matrix $\mathbf{B}^i_{<q,\{l,r\}}$ can be partitioned into block rows of heights $M_1, \ldots, M_{q-1}$ and the matrix and $\mathbf{B}^i_{<q,\{l,r\}}$ in block rows of heights $M_{q+1}, \ldots, M_K$ such that each block row corresponds to a submatrix of the filter matrix $\mathbf{W}$. Each block row can then be further partitioned into $|\mathcal{N}_q|$, $Q/2$-wide blocks. In each such block row, only one of those blocks is different from $\mathbf{0}$. If the block row corresponds to $\mathbf{W}_k$ and $k \in \mathcal{V}_{n_l q}$ with $n_o \in \mathcal{N}_q$ and $f_q(n_o) = o$, then the o-th block matrix of the k-th block row of $\mathbf{B}^i_{<q,\{l,r\}}$ if $k <$ or $\mathbf{B}^i_{<q,\{l,r\}}$ if $k > q$ is equal to $\mathbf{W}^i_{k,\{l,r\}}$.

Then indeed

$${\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{x}_{\{l,r\}} = \widetilde{\mathbf{x}}^i_{q,\{l,r\}} \,, \tag{3.29}$$

such that with these new definitions of the parametrization and of $\widetilde{\mathbf{x}}^i_{k,\{l,r\}}$, the compressed covariance matrices can be defined similar to the previous sections' and the optimization problem in each iteration can be rewritten as a problem of the same form but in terms of the compressed and local signals only. The resulting algorithm is then analogous to that of the previous section but uses the new definitions of the variables and a more elaborate data flow as presented in this section, and is given in Algorithm 3.3.

Algorithm 3.3: t-DACGEE

```
Set $i \leftarrow 0,\ q \leftarrow 1$ and initialize all $\mathbf{W}^0_{k,\{l,r\}}\ \forall k \in \mathcal{K}$ as random matrices
repeat
    Each node $k \in \mathcal{N}$ transmits $N$ compressed observations $\mathbf{z}^i_{k\to n_l,\{l,r\}}$ and
    $\mathbf{z}^i_{k\to n_l,\{r,l\}}$ (as defined in eq. (3.22)) to node $n_l$, $\forall n_l \in \mathcal{N}_k$
    At node $q$:
      • Estimate $\widetilde{\mathbf{R}}_{q,\{l,r\}}$ and $\widetilde{\mathbf{R}}_{q,\{r,l\}}$ based on (3.23)
      • Solve
            $$\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \underset{\widetilde{\mathbf{W}}_q}{\arg\max}\ \mathrm{Tr}\left(\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}_{q,\{l,r\}} \widetilde{\mathbf{W}}_q\right)$$
            $$\text{s.t.}\quad \widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}_{q,\{r,l\}} \widetilde{\mathbf{W}}_q = \mathbb{I}$$
      • Resolve the sign ambiguity
            $$\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}} \mathbf{D}^*$$
            $$\text{where}\quad \mathbf{D}^* = \underset{\mathbf{D}\in\mathcal{D}}{\arg\min} \left\| \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} \mathbf{D} - \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}} \right\|_F$$
      • Define $P = |\mathcal{N}_q|\frac{Q}{2}$ and partition $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ as
            $$\mathbf{W}^{i+1}_{q,\{l,r\}} = \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$$
            $$\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{0} & \mathbb{I}_P \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$$
      • Take $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{G}_1^\top & \dots & \mathbf{G}_{\mathcal{N}_q}^\top \end{bmatrix}^\top$
      • Disseminate $\mathbf{G}_l$ over the branch $\mathcal{V}_{n_l q}$ for which $f_q(n_l) = l$, $\forall n_l \in \mathcal{N}_q$. Each
        node $k \in \mathcal{V}_{n_l q}$ updates
            $$\mathbf{X}^{i+1}_{k,\{l,r\}} = \mathbf{X}^{i}_{k,\{l,r\}} \mathbf{G}_l$$
    Set $i \leftarrow i+1$ and $q \leftarrow (q\ mod\ K) + 1$
end
```

Instead of constraining each filter submatrix' span (except for $\mathbf{W}^{i+1}_{q,\{l,r\}}$) to remain the same by multiplying each filter submatrix $\mathbf{W}^{i}_{k,\{l,r\}}$ with corresponding matrix $\mathbf{G}_k$, now all filter matrices corresponding to nodes within the same $\mathcal{V}_{n_o q}$ are multiplied with the same matrix $\mathbf{G}_o$. This reduces the degrees of freedom of the optimization problem in each iteration. Hence it is expected that the convergence of t-DACGEE will be slower than that of DACGEE.

Nevertheless, t-DACGEE preserves the advantageous properties of DACGEE: the

communication between neighbouring nodes is similarly reduces as long as $Q < M_k$ for node k. Furthermore, the compressed covariance matrices are recalculated in each iteration based on new observations and hence it can adapt to slowly changing statistics.
In Section 3.5, t-DACGEE will also be extended to re-use a same batch of observations efficiently to increase the tracking performance.

3.4 Regularization

In Section 2.2.4 the (MIMO) CSP objective was regularized to avoid overfitting. To this end, an l_2-penalty on the norm of the filters was added. However, this requires the complete filter matrix to be available in each iteration and in each node. Each node thus needs to keep track of the complete matrix which in a tree topology requires relaying which is known to be poorly scalable [44].

By using the same principle as was used in the derivations of DACGEE and t-DACGEE, i.e. AO combined with a clever re-parametrization, each node only needs to broadcast a ${}^{Q}\!/_{2} \times {}^{Q}\!/_{2}$ matrix in each iteration, rather than an $M_k \times {}^{Q}\!/_{2}$ submatrix of $\mathbf{W}^{i+1}_{\{l,r\}}$. The method to efficiently incorporate an l_2-norm of the filter matrix into the distributed optimization was adapted from the solution to a resembling problem proposed by Musluoglu et al. [46]. Here it serves to incorporate an orthonormality constraint $\mathbf{W}^\top\mathbf{W} = \mathbb{I}$ into an efficient distributed realization of a different problem, Trace Ratio Optimization, rather than simply the norm of the left-hand side of this constraint.

For ease of exposition, the derivation of this approach will first be worked out for fully connected topologies, i.e. regularizing DACGEE, in a first subsection. Next, this will then be extended to tree topologies.

3.4.1 Fully Connected Topologies

Recall from Section 2.2.4 the formulation of the regularized MIMO CSP optimization problem given in (2.34):

$$\mathbf{W}^*_{\{l,r\}} = \arg\max_{\mathbf{W}} \frac{\mathrm{Tr}\left(\mathbf{W}^\top\mathbf{R}_{\{l,r\}}\mathbf{W}\right)}{\mathrm{Tr}\left((1-\lambda)\cdot\mathbf{W}^\top\mathbf{R}_{avg.}\mathbf{W}\right) + \lambda\cdot\mathrm{Tr}\left(\mathbf{W}^\top\mathbf{W}\right)} \, .$$

Now analogous to Section 3.3.1, transform this problem into an AO procedure where the optimization variable $\mathbf{W}$ is split into submatrices $\mathbf{W}_k$ where the entries of these submatrices correspond to a node k. In each iteration, only the submatrix of $\mathbf{W}$ corresponding to node q can be freely chosen to maximize the objective and the span of the other submatrices must remain the same.

First observe that compared to the non-regularized version, the numerator is identical. Second, the parametrization of (3.3-3.9) are used to rewrite the denominator in iteration i:

$$\begin{aligned}
&\operatorname{Tr}\left((1-\lambda)\cdot \mathbf{W}^\top \mathbf{R}_{avg.}\mathbf{W}\right) + \lambda \cdot \operatorname{Tr}\left(\mathbf{W}^\top \mathbf{W}\right) && (3.30)\\
&= \operatorname{Tr}\left(\frac{(1-\lambda)}{2}\cdot \widetilde{\mathbf{W}}^\top {\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{R}_{\{l,r\}} \mathbf{C}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}\right)\\
&\quad + \operatorname{Tr}\left(\frac{(1-\lambda)}{2}\cdot \widetilde{\mathbf{W}}^\top {\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{R}_{\{r,l\}} \mathbf{C}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}\right) && (3.31)\\
&\quad + \lambda \cdot \operatorname{Tr}\left(\widetilde{\mathbf{W}}^\top \underbrace{{\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{C}^i_{q,\{l,r\}}}_{\triangleq \mathbf{K}^i_{q,\{l,r\}}} \widetilde{\mathbf{W}}\right)\\
&= \operatorname{Tr}\left(\frac{(1-\lambda)}{2}\cdot \widetilde{\mathbf{W}}^\top \widetilde{\mathbf{R}}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}\right)\\
&\quad + \operatorname{Tr}\left(\frac{(1-\lambda)}{2}\cdot \widetilde{\mathbf{W}}^\top \widetilde{\mathbf{R}}^i_{q,\{r,l\}} \widetilde{\mathbf{W}}\right) && (3.32)\\
&\quad + \lambda \cdot \operatorname{Tr}\left(\widetilde{\mathbf{W}}^\top \mathbf{K}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}\right)\\
&= \operatorname{Tr}\left((1-\lambda)\cdot \widetilde{\mathbf{W}}^\top \widetilde{\mathbf{R}}^i_{q,avg.} \widetilde{\mathbf{W}}\right) + \lambda \cdot \operatorname{Tr}\left(\widetilde{\mathbf{W}}^\top \mathbf{K}^i_{q,\{l,r\}} \widetilde{\mathbf{W}}\right) && (3.33)\\
&= \operatorname{Tr}\left(\widetilde{\mathbf{W}}^\top \left((1-\lambda)\cdot \widetilde{\mathbf{R}}^i_{q,avg.} + \lambda \cdot \mathbf{K}^i_{q,\{l,r\}}\right) \widetilde{\mathbf{W}}\right) , && (3.34)
\end{aligned}$$

where $\widetilde{\mathbf{R}}^i_{q,avg.} = \frac{\widetilde{\mathbf{R}}_{q,\{l,r\}}+\widetilde{\mathbf{R}}_{q,\{r,l\}}}{2}$.

The matrix $\mathbf{K}^i_{q,\{l,r\}}$ can be written as a block diagonal matrix:

$$\begin{aligned}
\mathbf{K}^i_{q,\{l,r\}} &= {\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{C}^i_{q,\{l,r\}} && (3.35)\\
&\overset{(3.6)}{=} \operatorname{blkdiag}\left(\mathbb{I}_{M_q},\ {\mathbf{B}^i_{<k,\{l,r\}}}^\top \mathbf{B}^i_{<k,\{l,r\}},\ {\mathbf{B}^i_{>k,\{l,r\}}}^\top \mathbf{B}^i_{>k,\{l,r\}}\right) && (3.36)\\
&\overset{(3.7\text{-}3.8)}{=} \operatorname{blkdiag}\left(\mathbb{I}_{M_q}, \mathbf{L}^i_{1,\{l,r\}}, \dots, \mathbf{L}^i_{q-1,\{l,r\}}, \mathbf{L}^i_{q+1,\{l,r\}}, \dots, \mathbf{L}^i_{K,\{l,r\}}\right) , && (3.37)
\end{aligned}$$

where

$$\mathbf{L}^i_{k,\{l,r\}} = {\mathbf{X}^i_{k,\{l,r\}}}^\top \mathbf{X}^i_{k,\{l,r\}} . \qquad (3.38)$$

Combining the numerator and denominator again, the compressed problem becomes

$$\begin{aligned}
\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \underset{\widetilde{\mathbf{W}}_q}{\arg\max}\ & \operatorname{Tr}\left(\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}_{q,\{l,r\}} \widetilde{\mathbf{W}}_q\right)\\
\text{s.t.}\quad & \widetilde{\mathbf{W}}_q^\top \left((1-\lambda)\cdot \widetilde{\mathbf{R}}^i_{q,avg.} + \lambda \cdot \mathbf{K}^i_{q,\{l,r\}}\right) \widetilde{\mathbf{W}}_q = \mathbb{I} .
\end{aligned} \qquad (3.39)$$

Observe that this is of the same form as in the non-regularized MIMO case of Section 2.2.2. Consequently, as was proven in that section, the solution can be found as the $Q/2$ largest generalized eigenvectors of the pair $\left(\widetilde{\mathbf{R}}_{q,\{l,r\}}, (1-\lambda)\cdot \widetilde{\mathbf{R}}^i_{q,avg.} + \lambda \cdot \mathbf{K}^i_{q,\{l,r\}}\right)$.

The extension of DACGEE to regularized CSP problems, hereafter referred to as r-DACGEE, thus requires that in each iteration each node $k \in \mathcal{K} \setminus \{q\}$, in addition to the compressed observations, transmits the matrix $\mathbf{L}^i_{k,\{l,r\}}$ to node q. The full algorithm is given in Alg. 3.4.

Algorithm 3.4: r-DACGEE

```
Set $i \leftarrow 0$, $q \leftarrow 1$ and initialize all $\mathbf{W}^0_{k,\{l,r\}}\ \forall k \in \mathcal{K}$ as random matrices
**repeat**
    Each node $k \in \mathcal{K}$ compresses and broadcasts $N$ new observations
      $\bar{\mathbf{x}}^i_{k,\{l,r\}} = {\mathbf{W}^i_{k,\{l,r\}}}^\top \mathbf{x}^i_{k,\{l,r\}}$ and $\bar{\mathbf{x}}^i_{k,\{r,l\}} = {\mathbf{W}^i_{k,\{r,l\}}}^\top \mathbf{x}^i_{k,\{r,l\}}$
    Each node $k \in \mathcal{K} \setminus \{q\}$ sends $\mathbf{L}^i_{k,\{l,r\}} = {\mathbf{X}^i_{k,\{l,r\}}}^\top \mathbf{X}^i_{k,\{l,r\}}$
    At node $q$:
      • Estimate $\widetilde{\mathbf{R}}_{q,\{l,r\}}$ and $\widetilde{\mathbf{R}}_{q,\{r,l\}}$
      • Solve
          $$\begin{aligned}\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} &= \underset{\widetilde{\mathbf{W}}_q}{\arg\max}\ \operatorname{Tr}\left(\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}_{q,\{l,r\}} \widetilde{\mathbf{W}}_q\right)\\ \text{s.t.}\quad & \widetilde{\mathbf{W}}_q^\top \left((1-\lambda)\cdot \widetilde{\mathbf{R}}^i_{q,avg.} + \lambda \cdot \mathbf{K}^i_{q,\{l,r\}}\right) \widetilde{\mathbf{W}}_q = \mathbb{I}\end{aligned}$$
      • Resolve the sign ambiguity
          $$\begin{aligned}\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} &= \widetilde{\mathbf{W}}^i_{q,\{l,r\}} \mathbf{D}^*\\ \text{where}\quad \mathbf{D}^* &= \underset{\mathbf{D}\in\mathcal{D}}{\arg\min} \left\| \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} \mathbf{D} - \widetilde{\mathbf{W}}^i_{q,\{l,r\}} \right\|_F\end{aligned}$$
      • Define $P = (K-1)Q$ and partition $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ as
          $$\begin{aligned}\mathbf{W}^{i+1}_{q,\{l,r\}} &= \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}\\ \mathbf{G}_{-q} &= \begin{bmatrix} \mathbf{0} & \mathbb{I}_P \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}\end{aligned}$$
      • Broadcast $\mathbf{G}_{-q}$ to all other nodes
    Each node $k \in \mathcal{K} \setminus \{q\}$ updates
          $$\mathbf{W}^{i+1}_{k,\{l,r\}} = \mathbf{W}^i_{k,\{l,r\}} \mathbf{G}_k$$
      where $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{G}_1^\top & \dots & \mathbf{G}_{q-1}^\top & \mathbf{G}_{q+1}^\top & \dots & \mathbf{G}_K^\top \end{bmatrix}^\top$
    Set $i \leftarrow i+1$ and $q \leftarrow (q\ mod\ K) + 1$
**end**
```

3.4.2 Tree topologies

As t-DACGEE is a generalizeation of DACGEE, the derivation of the previous subsection remains valid for tree topologies up until (3.35), so long as the variable definitions of Section 3.3.2 are used instead.

Using the new definition from (3.28) for $\mathbf{C}^i_{q,\{l,r\}}$, $\mathbf{K}^i_{q,\{l,r\}}$ now equates to (compare to (3.35)):

$$\begin{aligned}
\mathbf{K}^i_{q,\{l,r\}} &= {\mathbf{C}^i_{q,\{l,r\}}}^\top \mathbf{C}^i_{q,\{l,r\}} && (3.40)\\
&\overset{(3.28)}{=} \begin{bmatrix} \mathbb{I}_{M_q} & \\ & {\mathbf{B}^i_{<k,\{l,r\}}}^\top \mathbf{B}^i_{<k,\{l,r\}} + {\mathbf{B}^i_{>k,\{l,r\}}}^\top \mathbf{B}^i_{>k,\{l,r\}} \end{bmatrix} && (3.41)\\
&= \left[\begin{array}{c|ccc} \mathbb{I}_{M_q} & & & \\ \hline & \sum\limits_{k\in\mathcal{V}_{n_1 q}} \mathbf{L}^i_{k,\{l,r\}} & & \\ & & \ddots & \\ & & & \sum\limits_{k\in\mathcal{V}_{n_{\mathcal{N}_q} q}} \mathbf{L}^i_{k,\{l,r\}} \end{array}\right], && (3.42)
\end{aligned}$$

with $\mathbf{L}^i_{k,\{l,r\}}$ defined as in (3.38).

Now define $\mathbf{L}^i_{k\to l,\{l,r\}}$ as the matrix that node k sends to node l to be

$$\mathbf{L}^i_{k\to l,\{l,r\}} = \mathbf{L}^i_{k,\{l,r\}} + \sum_{m\in\mathcal{N}_k\setminus l} \mathbf{L}^i_{m\to k,\{l,r\}} \,. \tag{3.43}$$

Observe that similar to (3.24), this is equivalent to

$$\mathbf{L}^i_{l\to k,\{l,r\}} = \sum_{m\in\mathcal{V}_{lk}} \mathbf{L}^i_{m,\{l,r\}} \,. \tag{3.44}$$

Hence (compare to (3.37)):

$$\mathbf{K}^i_{q,\{l,r\}} = \mathrm{blkdiag}\left(\mathbb{I}_{M_q}, \mathbf{L}^i_{n_1\to q,\{l,r\}}, \dots, \mathbf{L}^i_{n_{\mathcal{N}_q}\to q,\{l,r\}}\right) . \tag{3.45}$$

This derivation thus shows that by constructing the matrix $\mathbf{K}^i_{q,\{l,r\}}$ with the $\mathbf{L}^i_{n_l\to q,\{l,r\}}$ rather than with the $\mathbf{L}^i_{k,\{l,r\}}$ as in the fully connected case, and by similarly constructing the compressed covariance matrices as explained in Section 3.3.2, the rest of the derivation of r-DACGEE can be straightforwardly applied to t-DACGEE to obtain regularized t-DACGEE, further referred to as rt-DACGEE.

The definition of $\mathbf{L}^i_{k\to l,\{l,r\}}$ in (3.43) also means that it can be calculated by the same fusion and diffusion process as for $\mathbf{z}^i_{k\to l,\{l,r\}}$ (compare to (3.22)). However only node q has a use for these matrices and thus a fusion flow with q as the root is sufficient. For notational simplicity however, this is ignored in the text.

The resulting algorithm for rt-DACGEE is given in Alg. 3.5.

Algorithm 3.5: rt-DACGEE

```
Set $i \leftarrow 0,\ q \leftarrow 1$ and initialize all $\mathbf{W}^0_{k,\{l,r\}}\ \forall k \in \mathcal{K}$ as random matrices
repeat
    Each node $k \in \mathcal{N}$ transmits $N$ compressed observations $\mathbf{z}^i_{k\to n_l,\{l,r\}}$ and
      $\mathbf{z}^i_{k\to n_l,\{r,l\}}$ (as defined in eq. (3.22)) to node $n_l$, $\forall n_l \in \mathcal{N}_k$
    Each node $k \in \mathcal{N}$ transmits $\mathbf{L}^i_{k\to n_l,\{l,r\}}$ (as defined in eq. (3.43)) to node
      $n_l$, $\forall n_l \in \mathcal{N}_k$
    At node $q$:
      • Estimate $\widetilde{\mathbf{R}}_{q,\{l,r\}}$ and $\widetilde{\mathbf{R}}_{q,\{r,l\}}$ based on (3.23)
      • Solve
          $$\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \underset{\widetilde{\mathbf{W}}_q}{\arg\max}\ \mathrm{Tr}\left(\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}_{q,\{l,r\}} \widetilde{\mathbf{W}}_q\right)$$
          $$\text{s.t.}\quad \widetilde{\mathbf{W}}_q^\top\left((1-\lambda)\cdot\widetilde{\mathbf{R}}^i_{q,avg.} + \lambda\cdot\mathbf{K}^i_{q,\{l,r\}}\right)\widetilde{\mathbf{W}}_q = \mathbb{I}$$
      • Resolve the sign ambiguity
          $$\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \widetilde{\mathbf{W}}^i_{q,\{l,r\}}\mathbf{D}^*$$
          $$\text{where}\quad \mathbf{D}^* = \underset{\mathbf{D}\in\mathcal{D}}{\arg\min}\left\|\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}\mathbf{D} - \widetilde{\mathbf{W}}^i_{q,\{l,r\}}\right\|_F$$
      • Define $P = |\mathcal{N}|_q \frac{Q}{2}$ and partition $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ as
          $$\mathbf{W}^{i+1}_{q,\{l,r\}} = \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix}\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$$
          $$\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{0} & \mathbb{I}_P \end{bmatrix}\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$$
      • Take $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{G}_1^\top & \dots & \mathbf{G}_{\mathcal{N}_q}^\top \end{bmatrix}^\top$
      • Disseminate $\mathbf{G}_l$ over the branch $\mathcal{V}_{n_l q}$ for which $f_q(n_l) = l$, $\forall n_l \in \mathcal{N}_q$. Each
        node $k \in \mathcal{V}_{n_l q}$ updates
          $$\mathbf{X}^{i+1}_{k,\{l,r\}} = \mathbf{X}^i_{k,\{l,r\}}\mathbf{G}_l$$
    Set $i \leftarrow i+1$ and $q \leftarrow (q\ mod\ K) + 1$
end
```

3.5 Efficient data re-use

As touched upon at the end of Section 3.3.1, the DACGEE algorithm and its variants are adaptive algorithms. Indeed, as they use a new window of observations in each iteration and re-estimate the compressed covariance matrices from those windows, the algorithm inherently tracks (slowly) varying statistics through the re-estimation of the covariance matrices and consequently the step taken with the next iterate tracks the updated optimum.

Improved tracking speed

Yet, the iterative nature of the algorithm also means that the optimum is not reached in a single iteration. If the statistics vary too quickly, the convergence speed might be such that the iterate is not able to come close to the optimum before that optimum has already shifted away [47].

However for a fixed objective, it was proven that DACGEE converges to the optimum as calculated by central algorithm (i.e. from the GEVD of the network-wide covariance matrices). Therefore, if the number of iterations per window of data (which indeed has a fixed objective as the covariance matrices are only re-estimated per window) is increased, so is the speed of the tracking.

Figure 3.3 shows the impact on the convergence speed of DACGEE for the example on artificially generated data as in the original paper [20] for both fully connected and tree topologies. As R is increased, the x-axis is essentially compressed and the time it takes to reach a certain error is inversely proportional with R. Observe the difference in the scale of the x-axis between plots 3.3a and 3.3b: as hypothesized in Section 3.1 the convergence is indeed slower due to the partial connections in the tree topology.

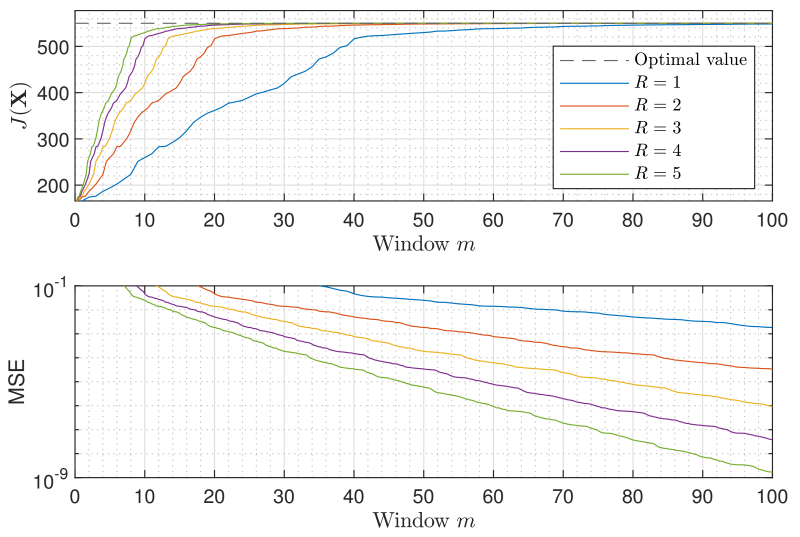
Each iteration requires observations compressed with the most recent filter matrix. Consequently, after an update of the filter matrix, the nodes need to update their part of the matrix locally and then compress the same window of data again but now with the new filters, and transmit them to the updating node.
Hence a single window of observations is compressed and subsequently transmitted multiple times in order to be able to perform multiple iterations of the DACGEE algorithm per window.

Observe that those multiple iterations must be done on different updating nodes q. Indeed, for the same updating node there is no extra flexibility and if there exist matrices $\mathbf{G}_k$ that would further optimize the problem for the same updating node q, then $\widetilde{\mathbf{W}}^i_{q,\{l,r\}}$ was not optimal in the first update.

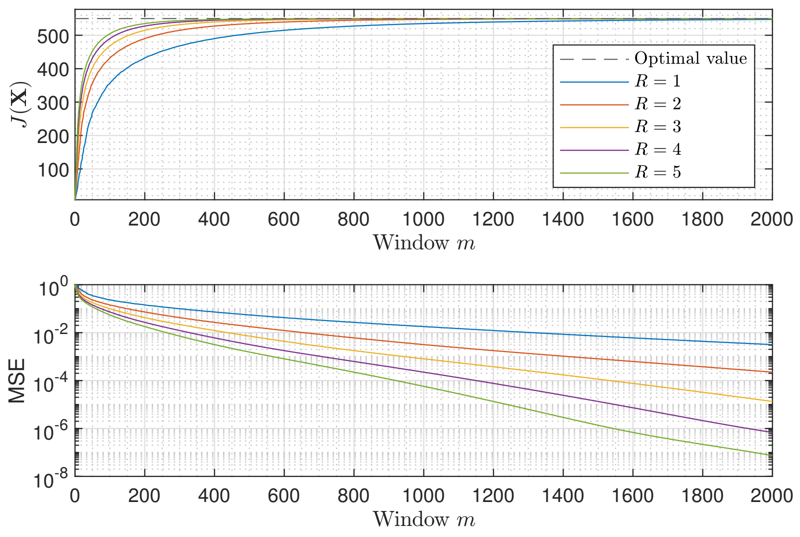
Scaling behaviour

Because of the retransmission, the energy consumption per window of data is a multiple of that of the standard DACGEE algorithm. In fact, if there are R transmissions of the same observations (albeit compressed using different filters in each iteration), it is easy to see that the energy consumption will be R times higher as simply multiple iterations of the complete DACGEE algorithm are performed per window. The amount of iterations per window R will further be referred to in this text as the *re-use factor*.

To mitigate this poor scaling behaviour in fully connected topologies, an efficient



(A) Fully Connected topology (DACGEE)



(B) Tree topology (tDACGEE)

FIGURE 3.3: Impact of re-use on the convergence speed in DACGEE and tDACGEE on synthetic data as in the example in [20]. The upper plot shows the evolution of the value of the CSP objective function, the lower plot the mean squared error on the entries of the iterate w.r.t. the optimal solution. For both subplots, the same random data was used on a 253-channel topology clustered into $K = 40$ nodes based on their location. The number of filters was set to $Q = 2$. In each iteration, the same 2000 observations were used from either class as generated by the procedure from [20].

approach was proposed by Musluoglu et al. [47]. The derivation of this algorithm is the topic of a first subsection. Next, continuing the approach taken in [47], the method will be extended to tree topologies in a second subsection.

3.5.1 Fully Connected topologies

Rather than retransmit the same set of observations after compression with the updated filters, the nodes update the already received compressed observations from nodes $\forall k \in \mathcal{K} \setminus \{q\}$ instead. This can be done as follows:

$$\bar{\mathbf{x}}_{k,\{l,r\}}^{i+1} = {W_{k,\{l,r\}}^{i+1}}^\top \mathbf{x}_{k,\{l,r\}}^{i} \tag{3.46}$$

$$= \mathbf{G}_k^\top {W_{k,\{l,r\}}^{i}}^\top \mathbf{x}_{k,\{l,r\}}^{i} \tag{3.47}$$

$$= \mathbf{G}_k^\top \bar{\mathbf{x}}_{k,\{l,r\}}^{i}\,, \tag{3.48}$$

where iterations i and $i+1$ correspond to the same window of observations, i.e. where it holds that

$$\left\lfloor \frac{i}{R} \right\rfloor = \left\lfloor \frac{i+1}{R} \right\rfloor = m\,.$$

Here, m denotes the index of the window.

For the updating node q, there is however no relation between $\mathbf{X}_{q,\{l,r\}}^{i}$ and $\mathbf{X}_{q,\{l,r\}}^{i+1}$ and hence the updating node does need to retransmit these signals after compressing with $\mathbf{X}_{q,\{l,r\}}^{i+1}$.

Hence rather than having each node re-transmit in every re-use of a window of data, only the updating node q incurs additional energy usage. As the updating node changes with each iteration (cf. supra), this additional energy usage is spread over time over all nodes, as will be discussed in the next chapter.

Similarly, if applicable, the $\mathbf{L}_{k,\{r,l\}}^{i}$ matrices of rDACGEE (alg. 3.4) need not be retransmitted either as they can similarly be updated:

$$\mathbf{L}_{l,\{l,r\}}^{i+1} = \mathbf{G}_l^\top \mathbf{L}_{l,\{l,r\}}^{i} \mathbf{G}_l\,. \tag{3.49}$$

Note that this requires that every node (or at least the $R-1$ next updating nodes) has the matrices $\mathbf{L}_{k,\{l,r\}}^{i}$ available to it in order to be able to update them in case it becomes an updating node.

The method described above then leads to algorithm 3.6, referred to as *r-DACGEE with re-use*. For completeness, re-use is demonstrated here as an extension to r-DACGEE rather than DACGEE itself. However note that the efficient re-use technique of this section can equally well be applied to DACGEE.

3.5.2 Tree topologies

The same idea can be applied to tree topologies with one caveat however due to the cumulation in the data flow.

As in the fully connected case, the updated filter matrix of the updating node q $\mathbf{W}^{i+1}_{q,\{l,r\}}$ again has no straightforward relation to $\mathbf{W}^{i}_{q,\{l,r\}}$ and therefore q has to recompress and retransmit its observations to its neighbours. However, as the signals $\{\mathbf{z}^{i+1}_{n_l\to n_m,\{l,r\}} | n_l \in \mathcal{N}_q \wedge n_m \in \mathcal{N}_{n_l} \setminus \{q\}\}$ depend on these signals $\{\mathbf{z}^{i+1}_{q\to n_l,\{l,r\}} | n_l \in \mathcal{N}_q\}$ the former set of signals needs to be transmitted to the child nodes of the neighbours of q as well. This pattern continues downward due to the fact that upstream signals are accumulated into the signals sent downstream. Hence unlike in the fully connected case, a retransmission is required at each level of the tree, creating a diffusion flow starting from node q. Indeed, due to the absence of strict criteria of what defines a tree, a tree can always be reinterpreted to have a different root and still comply to our definition of a tree (i.e. only 1 unique path between two nodes).

In the other direction, i.e. from the leaves of the q-rooted tree towards q, the signals are not dependant on those of q and hence the same updating can be used in each node k as in the fully connected case for those signals, albeit using the matrix $\mathbf{G}_l$ corresponding to the sub-branch of node q the node is in rather than the matrix $\mathbf{G}_k$ corresponding to the node itself:

$$\mathbf{z}^{i+1}_{n_l\to k,\{l,r\}} = \mathbf{G}_l^\top \mathbf{z}^{i}_{m\to k,\{l,r\}} \qquad \forall m \in \mathcal{N}_k \setminus \{n_{uplink}\} \text{ and } \forall k \in \mathcal{N}\,, \tag{3.50}$$

where it holds that $k \in \mathcal{V}_{n_l q}$, $f_q(n_l) = l$ and the uplink node n_{uplink} is defined as the neighbouring node of k that is on the path that connects k to q.

The updated matrices $\mathbf{L}^{i+1}_{k\to m,\{l,r\}}$ can similarly be obtained from the matrices $\mathbf{L}^{i}_{k\to m,\{l,r\}}$ in the same way: Matrices sent upstream towards q can be updated

Algorithm 3.6: r-DACGEE with re-use

```
Set $i \leftarrow 0$, $q \leftarrow 1$ and initialize all $\mathbf{W}^0_{k,\{l,r\}}$ $\forall k \in \mathcal{K}$ as random matrices
repeat
    for $r = 1, \dots, R$ do
        if $r = 1$ then
            Each node $k \in \mathcal{K}$ compresses and broadcasts $N$ new observations
              $\bar{\mathbf{x}}^m_{k,\{l,r\}} = {\mathbf{W}^i_{k,\{l,r\}}}^\top \mathbf{x}^i_{k,\{l,r\}}$ and $\bar{\mathbf{x}}^i_{k,\{r,l\}} = {\mathbf{W}^m_{k,\{r,l\}}}^\top \mathbf{x}^i_{k,\{r,l\}}$
            Each node $k \in \mathcal{K} \setminus \{q\}$ broadcasts $\mathbf{L}^i_{k,\{l,r\}} = {\mathbf{X}^i_{k,\{l,r\}}}^\top \mathbf{X}^i_{k,\{l,r\}}$
        end
        At node $q$:
            • Estimate $\widetilde{\mathbf{R}}_{q,\{l,r\}}$ and $\widetilde{\mathbf{R}}_{q,\{r,l\}}$
            • Solve
                $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \arg\max_{\widetilde{\mathbf{W}}_q} \operatorname{Tr}\left(\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}_{q,\{l,r\}} \widetilde{\mathbf{W}}_q\right)$
                s.t. $\widetilde{\mathbf{W}}_q^\top \left((1-\lambda)\cdot \widetilde{\mathbf{R}}^i_{q,avg.} + \lambda \cdot \mathbf{K}^i_{q,\{l,r\}}\right) \widetilde{\mathbf{W}}_q = \mathbb{I}$
            • Resolve the sign ambiguity
                  $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}}\mathbf{D}^*$
                  where $\mathbf{D}^* = \arg\min_{\mathbf{D}\in\mathcal{D}} \left\| \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}\mathbf{D} - \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}} \right\|_F$
            • Define $P = (K-1)Q$ and partition $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ as
                  $\mathbf{W}^{i+1}_{q,\{l,r\}} = \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$
                  $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{0} & \mathbb{I}_P \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$
            • Broadcast $\mathbf{G}_{-q}$ to all other nodes
            • if $r \neq R$ then Broadcast $\bar{\mathbf{x}}^{i+1}_{q,\{l,r\}} = \mathbf{W}^{i+1}_{q,\{l,r\}}\mathbf{x}^m_{q,\{l,r\}}$ and
              $\mathbf{L}^{i+1}_{k,\{l,r\}} = \mathbf{X}^{i+1}_{k,\{l,r\}}{}^\top \mathbf{X}^{i+1}_{k,\{l,r\}}$ to all other nodes
        Each node $k \in \mathcal{K} \setminus \{q\}$ updates
                  $\mathbf{W}^{i+1}_{k,\{l,r\}} = \mathbf{W}^{i}_{k,\{l,r\}}\mathbf{G}_k$
        where $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{G}_1^\top & \dots & \mathbf{G}_{q-1}^\top & \mathbf{G}_{q+1}^\top & \dots & \mathbf{G}_K^\top \end{bmatrix}^\top$
        if $r \neq R$ then
            Each node $k \in \mathcal{N}$ updates the previously received observations
            $\bar{\mathbf{x}}^i_{l,\{l,r\}}$ and $\mathbf{L}^i_{l,\{l,r\}}$ $\forall l \in \mathcal{K} \setminus \{q\}$:
                  $\bar{\mathbf{x}}^{i+1}_{l,\{l,r\}} = \mathbf{G}_l^\top \bar{\mathbf{x}}^i_{l,\{l,r\}}$
                  $\mathbf{L}^{i+1}_{l,\{l,r\}} = \mathbf{G}_l^\top \mathbf{L}^i_{l,\{l,r\}}\mathbf{G}_l$
        end
        $i \leftarrow i+1$, $q \leftarrow (q \mod K) + 1$
    end
    $m \leftarrow m+1$
end
```

from the previous iteration i as:

$$\mathbf{L}^{i+1}_{n_l \to q,\{l,r\}} = \mathbf{G}_l^\top \mathbf{L}^i_{n_l \to q,\{l,r\}}\mathbf{G}_l \,, \tag{3.51}$$

where $f_q(n_l) = l$. Matrices diffused from q need to retransmitted.

Observe that as the updating node q switches with each iteration, it is necessary that every node (or at least the $R-1$ next updating nodes) has all matrices $\mathbf{L}$ available to it in order to be able to update them in case it becomes an updating node.

Furthermore, as the signals and matrices are combinations of those from sub-branches, the interconnections in the network must stay the same in order to be able to update the already received observations. Hence for tree topologies, only static networks can be used and not dynamic networks. In practice this necessitates a network protocol that uses a static topology and if a link were to fail, builds a new topology and alerts the feature extraction algorithm that previous observations cannot be re-used in the next iteration.

Although the **L** matrices can be updated in the way described above, it is less efficient than performing a fusion flow to node q only in each iteration. Indeed, in a fusion flow, each node (except for q) does only 1 transmission to its uplink. In a diffusion flow however – as indeed required for updating the matrices **L** – each node (except for the leaves) must transmit to each of its children as well as requiring extra computations afterwards to do the updating (although this will turn out to be negligible, see Section 4.1).

Combining all of the above yields Algorithm 3.7, referred to as *rtDACGEE with re-use*. Again the same remark as in the previous section holds: re-use is demonstrated as an extension to the regularized version of tDACGEE although re-use can equally well be applied without regularization by skipping the steps in the algorithm involving the **L** matrices.

Algorithm 3.7: rt-DACGEE with re-use

```
Set $i \leftarrow 0,\ q \leftarrow 1$ and initialize all $\mathbf{W}^0_{k,\{l,r\}}\ \forall k \in \mathcal{K}$ as random matrices
repeat
    for $r = 1, \ldots, R$ do
        if $r = 1$ then
            Each node $k \in \mathcal{N}$ transmits $N$ compressed observations $\mathbf{z}^i_{k\to n_l,\{l,r\}}$
              and $\mathbf{z}^i_{k\to n_l,\{r,l\}}$ (as defined in eq. (3.22)) to node $l$, $\forall n_l \in \mathcal{N}_k$
        end
        Each node $k \in \mathcal{K} \setminus \{q\}$ transmits $\mathbf{L}^i_{k\to n_{uplink},\{l,r\}}$ (as defined in (3.43))
          to node $n_{uplink}$, the node that links node $k$ to node $q$
        At node $q$:
            • Estimate $\widetilde{\mathbf{R}}_{q,\{l,r\}}$ and $\widetilde{\mathbf{R}}_{q,\{r,l\}}$ based on (3.23)
            • Solve
              $$\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \underset{\widetilde{\mathbf{W}}_q}{\arg\max}\ \mathrm{Tr}\left(\widetilde{\mathbf{W}}_q^\top \widetilde{\mathbf{R}}_{q,\{l,r\}} \widetilde{\mathbf{W}}_q\right)$$
              $$\text{s.t.}\quad \widetilde{\mathbf{W}}_q^\top \left((1-\lambda)\cdot \widetilde{\mathbf{R}}^i_{q,avg.} + \lambda \cdot \mathbf{K}^i_{q,\{l,r\}}\right) \widetilde{\mathbf{W}}_q = \mathbb{I}$$
            • Resolve the sign ambiguity
                  $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}}\mathbf{D}^*$
                  where $\mathbf{D}^* = \arg\min_{\mathbf{D}\in\mathcal{D}} \left\|\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}\mathbf{D} - \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}}\right\|_F$
            • Define $P = |\mathcal{N}|_q \frac{Q}{2}$ and partition $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$ as
                  $\mathbf{W}^{i+1}_{q,\{l,r\}} = \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$
                  $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{0} & \mathbb{I}_P \end{bmatrix} \widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}$
            • Take $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{G}_1^\top & \dots & \mathbf{G}_{\mathcal{N}_q}^\top \end{bmatrix}^\top$
            • Disseminate $\mathbf{G}_l$ over the branch $\mathcal{V}_{n_l q}$ for which $f_q(n_l) = l$, $\forall n_l \in \mathcal{N}_q$. Each
              node $k \in \mathcal{V}_{n_l q}$ updates
                  $\mathbf{X}^{i+1}_{k,\{l,r\}} = \mathbf{X}^{i}_{k,\{l,r\}}\mathbf{G}_l$
            • **if** $r \neq R$ **then** Update the received compressed observations
              $\mathbf{z}^i_{n_l\to q,\{l,r\}} \forall n_l \in \mathcal{N}_q$:
                  $\mathbf{z}^{i+1}_{n_l\to q,\{l,r\}} = \mathbf{G}^\top_{l,\{l,r\}}\mathbf{z}^i_{n_l\to q,\{l,r\}}$
              where $f_q(n_l) = l.$ ;
            • **if** $r \neq R$ **then** Transmit $\mathbf{z}^{i+1}_{q\to n_l,\{l,r\}}$ to node $n_l$;
        **if** $r \neq R$ **then**
            Each node $k \in \{n \mid n \in \mathcal{V}_{n_l q} \wedge n_l \in \mathcal{N}_q\}$ updates the received
            compressed observations $\mathbf{z}^i_{o\to k,\{l,r\}}$, $\forall o \in \mathcal{N}_k \setminus \{n_{k,uplink}\}$:
                  $\mathbf{z}^{i+1}_{o\to k,\{l,r\}} = \mathbf{G}^\top_{l,\{l,r\}}\mathbf{z}^i_{o\to k,\{l,r\}}$
            where $f_q(n_l) = l$ and $n_{k,uplink}$ is the node through which node $k$
            connects to node $q$.
            Diffuse to all nodes $o \in \mathcal{N}_k \setminus \{n_{k,uplink}\}$: $\mathbf{z}^{i+1}_{k\to o,\{l,r\}}$
        **end**
        Set $i \leftarrow i + 1$ and $q \leftarrow (q \ mod \ K) + 1$
    **end**
    Set $m \leftarrow m + 1$
**end**
```

3.6 Integration into a classification pipeline

After deriving the distributed feature extraction algorithms in the previous sections, this section now considers this distributed feature extraction as part of an efficient MI classification pipeline for running on a WESN.

Recall that the reason for developing such distributed feature extraction algorithms was the extensive communication required between nodes to use the available spatial information. In the proposed pipeline, it is only the feature extraction that (optimally) extracts spatial information. If the distributed algorithms of the previous sections are used, each node then has access to the filtered network-wide signals. Consequently, each node is able to calculate the average log-power features and predict the class.

Hence, the only step in the proposed pipeline that needs a distributed version to enable an efficient classification pipeline on a WESN is the feature extraction, for which the efficient algorithms have already been discussed in the previous sections. The windows used in for the iterative updating can then also be used as the windows for calculating the features, i.e. the log average power of the filter outputs (see Section 2.2).

Integrating DACGEE and its variants into the complete pipeline then yields the algorithms given in overview Table 3.1. As was the case in the pipeline of Section 3.2, in each window observations of only 1 class are available and hence the updating procedure of Section 2.2 has to be used here as well. Note however that re-transmitting all observations from the past $L-1$ windows included in the sliding window, compressed with the most recent filters, would require a lot of communication. Rather, they are buffered in each node (that needs to do updating) as they were originally compressed. For the algorithms with re-use, the past observations are used as they were compressed by the filters for cycle $r = R$.

	Without re-use	With re-use
Fully Connected	Alg. 3.8	Alg. 3.10
Tree	Alg. 3.9	Alg. 3.11

TABLE 3.1: Overview of the algorithms giving the different variants of the distributed classification framework: with/without re-use for fully connected/tree topologies

Algorithm 3.8: Distributed MI classification in a fully connected topology without data re-use

```
Set $i \leftarrow 0$, $q \leftarrow 1$, and initialize all $\mathbf{W}_k^0$, $\forall k \in \mathcal{K}$ with random entries.
repeat
    parfor node $k \in \mathcal{K}$ do
        if $k \neq q$ then
            Compute and broadcast $\mathbf{L}_k^i = {\mathbf{W}_k^i}^\top \mathbf{W}_k^i$
        end
        Acquire $N$ new observations $\mathbf{x}_k^i$ and apply a $8-26$ Hz bandpass filter
        Broadcast $N$ new compressed observations $\bar{\mathbf{x}}_k^i[j] = {\mathbf{W}_k^i}^\top \mathbf{x}_k^i$
        Calculate the features as $\mathbf{y}^i = \log\left(\frac{1}{N}\sum_{j=1}^{N}\sum_{l=1}^{K}\bar{\mathbf{x}}_l^i[j]^2\right)$
        Update the covariance matrices (see Section 2.2), corresponding to the
          correct label '$l$' or '$r$'
        Update the filters
        if $k = q$ then
            Solve the compressed GEVD problem to obtain the $Q/2$ principal
              eigenvectors of $(\mathbf{R}_{k,l}^i + \alpha\mathbf{K}_q^i), \mathbf{R}_{k,r}^i)$ and of $(\mathbf{R}_{k,r}^i + \alpha\mathbf{K}_q^i, \mathbf{R}_{k,l}^i)$ to
              form $\widetilde{\mathbf{W}}_q^{i+1} \in \mathbb{R}^{Q\times\widetilde{M}_q}$ with
              $\mathbf{K}_q^i = \text{blkdiag}(\mathbb{I}_{M_q}, \mathbf{L}_1^i, \ldots, \mathbf{L}_{q-1}^i, \mathbf{L}_{q+1}^i, \ldots, \mathbf{L}_K^i)$
            Resolve the sign ambiguity
                $$\widetilde{\mathbf{W}}_{q,\{l,r\}}^{i+1} = \widetilde{\mathbf{W}}_{q,\{l,r\}}^{i}\mathbf{D}^*$$
                $$\text{where} \quad \mathbf{D}^* = \arg\min_{\mathbf{D}\in\mathcal{D}} \left\|\widetilde{\mathbf{W}}_{q,\{l,r\}}^{i+1}\mathbf{D} - \widetilde{\mathbf{W}}_{q,\{l,r\}}^{i}\right\|_F$$
            Set $\mathbf{W}_q^{i+1} = \begin{bmatrix}\mathbb{I}_{M_q} & \mathbf{0}\end{bmatrix}\widetilde{\mathbf{W}}_q^{i+1}$
            Broadcast $\mathbf{G}_{-q} = \begin{bmatrix}\mathbf{0} & I_{(K-1)Q}\end{bmatrix}\widetilde{\mathbf{W}}_q^{i+1}$ to all other nodes
        else
            Update the filter using the received matrix $\mathbf{G}_{-q}$ as $\mathbf{W}_k^{i+1} = \mathbf{W}_k^i\mathbf{G}_k$
              where $\mathbf{G}_{-q} = \begin{bmatrix}\mathbf{G}_1^\top & \ldots & \mathbf{G}_{q-1}^\top & \mathbf{G}_{q+1}^\top & \ldots & \mathbf{G}_K^\top\end{bmatrix}^\top$
        end
    end
    $i \leftarrow i+1$, $q \leftarrow (q \mod K) + 1$
end
```

Algorithm 3.9: Distributed MI classification in a tree topology without data re-use

```
Set $i \leftarrow 0$, $q \leftarrow 1$, and initialize all $\mathbf{W}_k^0$, $\forall k \in \mathcal{K}$ with random entries.
repeat
    parfor node $k \in \mathcal{K}$ do
        if $k \neq q$ then
            Compute $\mathbf{L}_k^i = \mathbf{W}_k^{i\top} \mathbf{W}_k^i$
            Take $l_{uplink} \in \mathcal{N}_k$ to be the node through which node $k$ connects
              to node $q$
            Compute and transmit to node $l_{uplink}$:
              $\mathbf{L}_{k \to l_{uplink}}^i = \mathbf{L}_k^i + \sum_{n_l \in \mathcal{N}_k \setminus \{l_{uplink}\}} \mathbf{L}_{n_l \to k}^i$
        end
        Acquire $N$ new observations $\mathbf{x}_k^i$ and apply a $8-26$ Hz bandpass filter
        Compute $\bar{\mathbf{x}}_k^i = \mathbf{W}_k^{i\top} \mathbf{x}_k^i$
        foreach $n_l \in \mathcal{N}_k$ do
            Transmit $N$ new compressed observations
              $\bar{\mathbf{x}}_{k \to n_l}^i = \bar{\mathbf{x}}_k^i + \sum_{n_m \in \mathcal{N}_k \setminus \{n_l\}} \bar{\mathbf{x}}_{n_m \to k}^i$ to node $n_l$
        end
        Calculate the features as $\mathbf{y}^i = \log\left(\frac{1}{N}\sum_{j=1}^{N}\left(\sum_{n_l \in \mathcal{N}_k} \bar{\mathbf{x}}_{n_l \to k}^i[j] + \bar{\mathbf{x}}_k^i[j]\right)^2\right)$
        Update the covariance matrices (see Section 2.2), corresponding to the
          correct label '$l$' or '$r$'
        Update the filters
        if $k = q$ then
            Solve the compressed GEVD problem to obtain the $Q/2$ principal
              eigenvectors of $(\mathbf{R}_{k,l}^i + \alpha\mathbf{K}_q^i), \mathbf{R}_{k,r}^i)$ and of $(\mathbf{R}_{k,r}^i + \alpha\mathbf{K}_q^i, \mathbf{R}_{k,l}^i)$ to
              form $\widetilde{\mathbf{W}}_q^{i+1} \in \mathbb{R}^{Q \times \widetilde{M}_q}$ with
              $\mathbf{K}_q^i = \text{blkdiag}(I_{M_q}, \mathbf{L}_{n_1 \to q}^i, \ldots, \mathbf{L}_{n_{|\mathcal{N}_q|} \to q}^i)$
            Resolve the sign ambiguity
              $$\widetilde{\mathbf{W}}_{q,\{l,r\}}^{i+1} = \widetilde{\mathbf{W}}_{q,\{l,r\}}^i \mathbf{D}^*$$
              $$\text{where} \quad \mathbf{D}^* = \arg\min_{\mathbf{D} \in \mathcal{D}} \left\| \widetilde{\mathbf{W}}_{q,\{l,r\}}^{i+1} \mathbf{D} - \widetilde{\mathbf{W}}_{q,\{l,r\}}^i \right\|_F$$
            Set $\mathbf{W}_q^{i+1} = \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{W}}_q^{i+1}$
            Take $\mathbf{G}_{\to q} = \begin{bmatrix} \mathbf{0} & \mathbb{I}_{|N_q|Q} \end{bmatrix} \widetilde{\mathbf{W}}_q^{i+1} \triangleq \begin{bmatrix} \mathbf{G}_1^\top & \ldots & \mathbf{G}_{|\mathcal{N}_q|}^\top \end{bmatrix}^\top$ and
              disseminate $\mathbf{G}_l$ over the subtree $\mathcal{V}_{n_l q}$ for which it holds that
              $f_q(n_l) = l$, $\forall n_l \in \mathcal{N}_q$
        else
            Update the filter using the received matrix $\mathbf{G}_l$ for which it holds
              that $k \in \mathcal{V}_{n_l q}$ and $f_q(n_l) = l$, as $\mathbf{W}_k^{i+1} = \mathbf{W}_k^i \mathbf{G}_l$
        end
    end
    $i \leftarrow i + 1$, $q \leftarrow (q \mod K) + 1$
end
```

Algorithm 3.10: Distributed MI classification in a fully connected topology with data re-use

```
Set $i \leftarrow 0$, $m \leftarrow 0$, $q \leftarrow 1$, and initialize all $\mathbf{W}_k^0, \forall k \in \mathcal{K}$ with random entries.
repeat
    Each node $k \in \mathcal{K} \setminus \{q\}$ acquires $N$ new observations $\mathbf{x}_k^m$ and applies a
      $8 - 26$ Hz bandpass filter
    Each node $k \in \mathcal{K}$ broadcasts $N$ new compressed observations
      $\bar{\mathbf{x}}_k^i = {\mathbf{W}_k^i}^\top \mathbf{x}_k^m$
    for $r = 1, \ldots, R$ do
        parfor node $k \in \mathcal{K}$ do
            if $k \neq q$ then
                Compute and broadcast $\mathbf{L}_k^i = {\mathbf{W}_k^i}^\top \mathbf{W}_k^i$
            end
            Calculate the features as $\mathbf{y}^i = \log\left(\frac{1}{N}\sum_{j=1}^N \sum_{l=1}^K \bar{\mathbf{x}}_l^i[j]^2\right)$
            Update the covariance matrices (see Section 2.2), corresponding to
              the correct label 'l' or 'r'
            Update the filters
            if $k = q$ then
                Solve the compressed GEVD problem to obtain the $Q/2$
                  principal eigenvectors of $(\mathbf{R}_{k,l}^i + \alpha\mathbf{K}_q^i), \mathbf{R}_{k,r}^i)$ and of
                  $(\mathbf{R}_{k,r}^i + \alpha\mathbf{K}_q^i, \mathbf{R}_{k,l}^i)$ to form $\widetilde{\mathbf{W}}_q^{i+1} \in \mathbb{R}^{Q \times \widetilde{M}_q}$ with
                  $\mathbf{K}_q^i = \text{blkdiag}(\mathbb{I}_{M_q}, \mathbf{L}_1^i, \ldots, \mathbf{L}_{q-1}^i, \mathbf{L}_{q+1}^i, \ldots, \mathbf{L}_K^i)$
                Resolve the sign ambiguity
                    $\widetilde{\mathbf{W}}_{q,\{l,r\}}^{i+1} = \widetilde{\mathbf{W}}_{q,\{l,r\}}^{i} \mathbf{D}^*$
                    where $\mathbf{D}^* = \underset{\mathbf{D} \in \mathcal{D}}{\arg\min} \left\| \widetilde{\mathbf{W}}_{q,\{l,r\}}^{i+1} \mathbf{D} - \widetilde{\mathbf{W}}_{q,\{l,r\}}^{i} \right\|_F$
                Set $\mathbf{W}_q^{i+1} = \begin{bmatrix} \mathbb{I}_{M_q} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{W}}_q^{i+1}$
                Broadcast $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{0} & \mathbb{I}_{(K-1)Q} \end{bmatrix} \widetilde{\mathbf{W}}_q^{i+1}$ to all other nodes
                if $r \neq R$ then
                    Compute and broadcast $\bar{\mathbf{x}}_k^{i+1}[j] = {\mathbf{W}_k^{(i+1)}}^\top \mathbf{x}_k[mN + j]$ for
                      $j = 1, \ldots, N$
            else
                Update the filter using the received matrix $\mathbf{G}_{-q}$ as
                 $\mathbf{W}_k^{i+1} = \mathbf{W}_k^i \mathbf{G}_k$ where
                 $\mathbf{G}_{-q} = \begin{bmatrix} \mathbf{G}_1^\top & \dots & \mathbf{G}_{q-1}^\top & \mathbf{G}_{q+1}^\top & \dots & \mathbf{G}_K^\top \end{bmatrix}^\top$
            end
            if $r \neq R$ then
                Update the received compressed observations as
                 $\bar{\mathbf{x}}_l^{i+1}[j] = \mathbf{G}_k^\top \bar{\mathbf{x}}_l^i[j]$ for $\forall j = \{1, \dots, N\}$ and $\forall l \in \mathcal{K} \setminus \{q\}$
            end
        end
        $i \leftarrow i + 1$, $q \leftarrow (q \mod K) + 1$
    end
    $m \leftarrow m + 1$
end
```

Algorithm 3.11: Sliding-window adaptive dCSP in a tree topology with data-reuse

```
Set $i \leftarrow 0$, $m \leftarrow 0$, $q \leftarrow 1$, and initialize all $\mathbf{W}_k^0$, $\forall k \in \mathcal{K}$ with random entries.
repeat
    parfor node $k \in \mathcal{K}$ do
        Acquire $N$ new observations $\mathbf{x}_k^m$ and applies a $8 - 26$ Hz bandpass
          filter
        Compute $\bar{\mathbf{x}}_k^i = {\mathbf{W}_k^i}^\top \mathbf{x}_k^m$
        foreach $l \in \mathcal{N}_k$ do
            Transmit $N$ new compressed observations
              $\bar{\mathbf{x}}_{k\to l}^i[j] = \bar{\mathbf{x}}_k^i[j] + \sum_{n_l \in \mathcal{N}_k \setminus \{l\}} \bar{\mathbf{x}}_{n_l \to k}^i[j]$ to node $n_l$, where
              $j = 1, \ldots, N$
        end
    end
    for $r = 1, \ldots, R$ do
        parfor node $k \in \mathcal{K}$ do
            if $k \neq q$ then
                Compute $\mathbf{L}_k^i = {\mathbf{W}_k^i}^\top \mathbf{W}_k^i$
                Take $l_{uplink} \in \mathcal{N}_k$ to be the node through which node $k$
                  connects to node $q$
                Compute and transmit to node $l_{uplink}$:
                  $\mathbf{L}_{k\to l_{uplink}}^i = \mathbf{L}_k^i + \sum_{n_l \in \mathcal{N}_k \setminus \{l_{uplink}\}} \mathbf{L}_{n_l \to k}^i$
            end
            Calculate the features as
              $\mathbf{y}^i = \log\left(\frac{1}{N}\sum_{j=1}^{N}\left(\sum_{b_l \in \mathcal{N}_k} \bar{\mathbf{x}}_{n_l \to k}^i[j] + \bar{\mathbf{x}}_k^i[j]\right)^2\right)$
            Update the covariance matrices (see Section 2.2), corresponding to
              the correct label 'l' or 'r'
            Update the filters
            if $k = q$ then
                Solve the compressed GEVD problem to obtain the $Q/2$
                  principal eigenvectors of $(\mathbf{R}^i_{k,l} + \alpha\mathbf{K}^i_q), \mathbf{R}^i_{k,r})$ and of
                $(\mathbf{R}^i_{k,r} + \alpha\mathbf{K}^i_q, \mathbf{R}^i_{k,l})$ to form $\widetilde{\mathbf{W}}^{i+1}_q \in \mathbb{R}^{Q\times\widetilde{M}_q}$ with
                $\mathbf{K}^i_q = \text{blkdiag}(\mathbb{I}_{M_q}, \mathbf{L}^i_{1\to q}, \dots, \mathbf{L}^i_{|\mathcal{N}_q|\to q})$
                Resolve the sign ambiguity
                    $\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}} = \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}}\mathbf{D}^*$
                    where $\mathbf{D}^* = \arg\min_{\mathbf{D}\in\mathcal{D}} \left\|\widetilde{\mathbf{W}}^{i+1}_{q,\{l,r\}}\mathbf{D} - \widetilde{\mathbf{W}}^{i}_{q,\{l,r\}}\right\|_F$
                Set $\mathbf{W}^{i+1}_q = \begin{bmatrix}\mathbb{I}_{M_q} & \mathbf{0}\end{bmatrix}\widetilde{\mathbf{W}}^{i+1}_q$
                Take $\mathbf{G}_{\to q} = \begin{bmatrix}\mathbf{0} & \mathbb{I}_{|\mathcal{N}_q|Q}\end{bmatrix}\widetilde{\mathbf{W}}^{i+1}_q \triangleq \begin{bmatrix}\mathbf{G}^\top_1 & \dots & \mathbf{G}^\top_{|\mathcal{N}_q|}\end{bmatrix}^\top$ and
                  disseminate $G_l$ over the subtree $\mathcal{V}_{n_q}$ where $f_q(n_l) = l$, $\forall j \in \mathcal{N}_q$
                if $r \neq R$ then
                    Compute $\bar{\mathbf{x}}^{i+1}_q[j] = \mathbf{X}^{i+1\top}_q\mathbf{x}_q[mN + j]$ for $j = 1, \dots, N$
            else
                Update the filter using the received matrix $\mathbf{G}_{l_j}$ for which holds
                  that $k \in \mathcal{V}_{n_l q}$, as $\mathbf{W}^{i+1}_k = \mathbf{W}^i_k\mathbf{G}_l$ where $f_q(n_l) = l$
                if $r \neq R$ then
                    Update the own compressed observations with that same
                      matrix as $\bar{\mathbf{x}}^{i+1}_k[j] = \mathbf{G}^\top_l\bar{\mathbf{x}}^i_k[j]$ for $\forall j = \{1, \dots, N\}$
            end
            if $r \neq R$ then
                foreach $n_l \in \mathcal{N}_k \setminus \{p \in \mathcal{N}_k | p \textit{ connects } n_l \textit{ to } q\}$ do
                    Update the received compressed observations as
                    $\bar{\mathbf{x}}^{i+1}_{n_l\to k}[j] = \mathbf{G}^\top_l\bar{\mathbf{x}}^i_{n_l\to k}[j]$ for $\forall j = \{1, \dots, N\}$ and with $\mathbf{G}_l$
                    such that $n_l \in \mathcal{V}_{n_l q}$ and $f_q(n_l) = l$
                end
                foreach $n_l \in \mathcal{N}_k \setminus \{p \in \mathcal{N}_k | p \textit{ connects } n_l \textit{ to } q\}$ do
                    Diffuse the observations updated by $q$ as
                    $\bar{\mathbf{x}}^{i+1}_{k\to n_l}[j] = \bar{\mathbf{x}}^{i+1}_k[j] + \sum_{m\in\mathcal{N}_k\setminus\{n_l\}} \bar{\mathbf{x}}^{i+1}_{m\to k}[j]$, where
                    $j = 1, \dots, N$
                end
            end
        end
        $i \leftarrow i + 1$, $q \leftarrow (q \mod K) + 1$
    end
    $m \leftarrow m + 1$
end
```

Part II

Evaluation

Chapter 4

Modelling of system performance

As discussed in Section 1.4, the following objectives can be used to define the performance and assess feasibility of the system as envisioned in this text:

- Minimal energy consumption
- Minimal latency
- Maximal decoding accuracy

As no physical implementation of the proposed system is available on which these can be measured however, these objectives will have to be estimated. For the energy consumption and latency, this can be done by means of modelling. For the decoding accuracy, previously recorded experimental data will be used to emulate WESN's which are then processed by the proposed classification pipelines. This will need to be done for systems with each of the proposed pipelines, i.e. each of the pipelines specified in Section 3.6.

This chapter presents the modelling of the energy consumption and latency. To do so, two key metrics of the proposed pipelines will need to be determined first: the computational complexity and required communication bandwidth. The modelling of these metrics are the subject of Sections 4.1 and 4.2 respectively. Based on the models for these two metrics, the following sections will further develop models for the latency and energy objectives (Sections 4.3 and 4.4 respectively).

In the next chapter, the models presented in this chapter will be evaluated on experimental data, together with simulations of the pipelines themselves to assess the accuracy of each pipeline and thus evaluate all three objectives. The results will then be used to compare the performance of the different pipelines as well as assessing the feasibility of the proposed approach altogether.

4.1 Computational complexity

For each of the pipelines presented in the previous chapter, the required amount of computations N_{comp} can be determined by determining the computational complexity of each of the steps and summing. Since in a practical implementation, the data is arbitrarily long, the complexity is determined per window, which has a finite duration.

As an example of how to determine the computational complexity, consider the first step of the centralized pipeline (Alg. 3.1) that requires computations: filtering the received observations. The observations in the window corresponding to iteration i, $\{\mathbf{x}^i[j]\}_{j\in\{1,\ldots,N\}}$, can be arranged into an $M \times N$ matrix $\mathbf{U}^i$, while the filter matrix $\mathbf{X}^i$ is of size $M \times Q$. The filtering operation can then be represented as the matrix multiplication ${\mathbf{X}^i}^\top \mathbf{U}^i$ which requires QMN multiplications and a similar amount of summations. Therefore, the computational complexity of this step is QMN FLOPS.

In the same fashion, the complexity of the other steps in the pipeline can be determined, which are given in Table A.1. The complexity result for the symmetric GEVD was taken from [49] since the algorithm used to calculate this decomposition is outside the scope of this text. Also shown in the table is the total complexity of one iteration i which is the sum of the complexities of the steps in the pipeline.

To improve insight in the complexity of the algorithms and to more easily make comparisons, the total complexity can be simplified by considering typical network parameters as given in Table 4.1. Compared to the calculation of the optimal filter ($\approx 16.6 \cdot 10^6$ FLOPS), the updating of the covariance matrices ($\approx 65.0 \cdot 10^6$ FLOPS) and the filtering ($\approx 1.0 \cdot 10^6$ FLOPS); the calculation of the features ($\approx 4.0 \cdot 10^3$ FLOPS) and resolving the sign ambiguity ($\approx 1.0 \cdot 10^3$ FLOPS) require more than 2 orders of magnitude less operations and can therefore be neglected. This simplifies the complexity model to $M^3 + NM^2 + QMN$ FLOPS.

Parameter	Typical value
N	2000
M	20
K	6
M_k	3
n_k	2
Q	4
R	5

Table 4.1: Typical network parameters

In a similar fashion the computational complexity can be modelled and then simplified for the other proposed pipelines as well.
For the distributed pipelines however, the steps differ slightly for the node that

is solving the compressed problem in a certain iteration as compared to the other nodes. Since in each iteration, a different node is the updating node and thus each node will eventually have done both tasks during operation, the mean complexity (over iterations and thus time) can be considered instead, without losing physical interpretation of the result. To weigh the two possibles cases (updating or non-updating), the relative frequency of each case is used. By using the mean complexity, a single result is obtained which makes it easier to compare pipelines to one another.

Furthermore, the Pipelines 3.10 and 3.11 make an extra distinction w.r.t. how many times the window of data corresponding to a certain iteration has already been used in order to more efficiently re-use that window of data for another update (see Section 3.5). Once again, using the relative frequency of a node to be in a particular case, the mean complexity can be considered instead as in general all nodes will have been in all possible cases during operation.

Since these pipelines re-use a unique data window R times, the mean complexity should be calculated per unique window – which has a fixed duration for a set sampling rate – rather than per iteration in order to fairly compare the pipelines to one another for the same sampling rate and window length. Indeed, the sampling rate and window length will limit how long the processing may take, and thus how much processing, can be done per window before incurring extra latency (see Section 4.3).

The full derivation (per step of the pipeline), simplification and calculation of the mean complexity are presented in detail in Appendix A.1 for all proposed pipelines. For the sake of easy exposition, the models were simplified even further than was done in the example above, by deriving simplified, approximate upper bounds. An overview of the final results of the mean complexity estimations per unique data window is given here in Table 4.2.

Pipeline	Complexity
Centralized (Alg. 3.1)	$M^3 + LM^2N + QMN$
FC no re-use (Alg. 3.8)	$L\widetilde{M_k}^2 N$
Tree no re-use (Alg. 3.9)	$L\widetilde{M_k}^2 N + QN\left(M_k + \lvert\mathcal{N}_k\rvert\right)$
FC with re-use (Alg. 3.10)	$LR\widetilde{M_k}^2 N + RKQ^2N$
Tree with re-use (Alg. 3.11)	$LR\widetilde{M_k}^2 N + RQN\Big(2M_k + \lvert\mathcal{N}_k\rvert^2 + \lvert\mathcal{N}_k\rvert + \lvert\mathcal{N}_k\rvert Q\Big)$

TABLE 4.2: Overview of the approximate upper bound on the mean computational complexity of the different proposed pipelines per window and per node

4.2 Communicational complexity

For each of the pipelines presented in the previous chapter, the communicational complexity can be estimated by counting the amount of words of data N_{comm} that need to be sent. That amount of words is equal to the sum of the number of entries of each of the variables that needs to be sent. Due to the same reason as in the previous section, this is done per window of data rather than per iteration for the pipelines with re-use (Algs. 3.10 and 3.11).

As an example of how to estimate the communicational complexity, consider Pipeline 3.1 and in particular the very first step (after initialization): collecting all measurements from the peripheral nodes in the central node.
Each node $k \in \mathcal{K} \setminus \{f_{FCe}\}$ acquires N observations for each of its M_k channel and hence needs to transmit M_kN words to the fusion centre, node k_{FCe}. Consequently, the fusion centre receives $\sum_{k \in \mathcal{N}} M_kN = MN$ words from the other nodes in the network.
In the same fashion, the communicational complexity of the other steps in the pipeline can be determined, which are given in Table A.6.

For the other algorithms, the communicational complexity can be determined in the same fashion. Similar to the previous section, the mean required bandwidth is used instead for these distributed pipelines, using the (same) relative frequency of the specific cases to calculate the mean. This makes the results easier to compare without losing the physical interpretation since each node will fulfil each role for a portion of the time given by that relative frequency. Furthermore, the results are similarly simplified even further by using simpler upper bounds to make the results more clear. For the pipelines with re-use, the complexity is given per window due to the same reason as was discussed in the previous section. The full derivations are presented in Appendix A.2 and an overview of the final results is given here in Table 4.3.

Pipeline	Transmission	Reception
Centralized (Alg. 3.1)	$\begin{cases} 0 & \text{if } k = c \\ NM_c & \text{if } k \neq c \end{cases}$	$\begin{cases} (M - M_c)N & \text{if } k = c \\ 0 & \text{if } k \neq c \end{cases}$
FC (Alg. 3.8)	NQ	KNQ
TI (Alg. 3.9)	$\lvert\mathcal{N}_k\rvert NQ$	$\lvert\mathcal{N}_k\rvert NQ$
FC with data re-use (Alg. 3.10)	$\left(\frac{R}{K} + 1\right) NQ + 2RQ^2$	$(R + K)NQ$
TI with data re-use (Alg. 3.11)	$\lvert\mathcal{N}_k\rvert RNQ$	$(R + \lvert\mathcal{N}_k\rvert)NQ$

TABLE 4.3: Overview of the approximate upper bounds on the mean communicational complexity of the different proposed algorithms per window and per node

4.3 Latency

In Section 1.4, the latency was defined as the time it takes for an input sample to have an effect on the (decoder) output.

The first subsection derives an estimate of that input-output latency for the algorithms proposed in Chapter 3, based on the models of Sections 4.1 and 4.2 to quantify the latency incurred by the algorithm between sample acquisition and output production.

Aside from producing an output, the algorithms proposed in Chapter 3 also adapt the covariance matrices and filters. The processing delay this introduces, hereafter called 'update processing delay' decreases the allowable input-output latency if the system is to complete its processing before a new window is completed. This is discussed in Subsection 4.3.2.

4.3.1 Input-output latency

In the centralized pipeline, only the fusion centre has all the observations available to it and hence is the only one that can decode the observations. Furthermore, as the fusion centre must sequentially receive the observations from all the peripheral nodes and can only then start its processing, it will have the largest latency of any node in the network. Here the latency is thus only considered for the fusion centre k_{FCe}.

In the case of the proposed distributed algorithms, each node is capable of decoding (see Chapter 3) and the latency hence needs be determined per node. However, these algorithms are synchronous in that sampling is done using the same clock and that windows of different nodes coincide. Thus, the maximum latency will determine when the network can continue processing. To determine the latency $T_{latency}$ for the pipelines, consider the steps done between acquiring an input sample and producing an output for a window of data:

Step 1: Acquisition
Step 2: Filtering
Step 3: Transmission/Receiving
  Step 3.1: Combination into neighbour specific signals
Step 4: Feature calculation

Before the window can be processed, all of the samples in that window need to be acquired[1]. For a window length of N samples and a sample rate of $f_s = {}^1/_{T_s}$ Hz, processing of the first sample will only start after $T_{window} = NT_s$ seconds.

The time it takes to filter the data can be calculated as the division of the number of operations that it requires $N_{comp,filtering}$ by the instruction rate of the processor (given in ${}^{instructions}/_{second} = IPS$). The former was already determined in the context of the computational complexity models discussed in Section 4.1 and can be found in Section A.1 for each of the pipelines. Similarly, the duration of Step 4 can be calculated in the same way.

This results in the following equation to calculate the latency:

$$\begin{aligned} T_{latency} &= T_{window} + T_{comp} + T_{comm} && (4.1) \\ &= NT_s + \frac{N_{comp,filtering} + N_{comp,features}}{IPS} + T_{comm}\,. && (4.2) \end{aligned}$$

The time it takes to do internode communication T_{comm} (Step 3) is dependant on the amount of words that each node needs to transmit and receive (as given by the models of Section 4.2) and the bandwidth of the communication channel, but also on the time it takes to synchronize with other nodes so that both nodes are ready to communicate. Indeed, a node can only transmit data to another node for which it already has the required data from other nodes. This may require a node to wait on another node.
Synchronization implies that some blocks of actions must be done sequentially. This limits the available amount of parallelism that can be exploited by the distributed nature of the system to reduce the latency. It is thus critical to schedule the communication between nodes such that parallel running blocks of actions in-between synchronization points are balanced in duration such that minimal time is spent waiting on another node.
Furthermore, if the order is not carefully chosen, deadlock situations may occur.

[1]This assumes that filtering of the observations will be done on the complete window of data at once, which is the case if the filtering is done on a DSP processor with SIMD instructions with sufficient bandwidth. If the filtering were to be implemented in the acquisition hardware, it would incur virtually no latency at all.

Thus in order to determine the total time it will take to do all communication, including both the time to do the transmission as well as the time spent waiting, the ordering of all communications must be specified such that the waiting time can be estimated as well as the time needed purely for transmission of the words. This can be done by means of a *communication schedule.*

Note that it will further be assumed that a node can only transmit to or receive from 1 other node at once and that collision can completely be avoided (a.o. by protocol, setting the transmission power as low as possible or using different channels).

For the centralized pipeline (Alg. 3.1), the following schedule is straightforward: nodes $1, \ldots, c-1, c+1, \ldots, K \in \mathcal{K}$ take turns transmitting their (uncompressed) data to the central node c. As the central node c needs to receive data from all peripheral nodes before it can start processing and is the only node that needs to do processing, its latency will be the largest and will decide the latency for the whole network. In Section 4.2 a method was proposed to determine the amount of words each node needs to transmit and receive. In Table A.6b this method was used to determine how many words the central node needs to receive. As all these words can be received back-to-back, this number together with the speed of the communication link BW [$^{bits}/_{s}$], can be used to determine the duration of the communication:

$$T_{comm,central} = k\frac{(M - Mc)\,N}{BW}\,, \tag{4.3}$$

with k the word size [$bits$]. For this algorithm, (4.2) then becomes (taking N_{comp} from Table A.1):

$$T_{latency,central} = NT_s + \frac{QMN + QN}{IPS} + k\frac{(M - Mc)\,N}{BW}\,. \tag{4.4}$$

For Pipelines 3.8 and 3.10, the following schedule can be used so that all nodes are able to calculate the output afterwards: all nodes take turns broadcasting their compressed observations to all other nodes while the other nodes are listening to the broadcasts. This will incur the same communication latency in all nodes, i.e.:

$$T_{comm,FC} = k\frac{(K-1)NQ + NQ}{BW} = k\frac{KNQ}{BW}\,, \tag{4.5}$$

where the information on the number of received and transmitted words was taken once again from the communication models of Section 4.2 (derivations in Appendix A, Tables A.7 and A.9). Note that with data re-use, only the first iteration per window is used (hence using filters optimized on data previous to the current window only) to keep the latency as short as possible.
The total latency (taking values for N_{comp} from Tables A.2 and A.4) is then:

$$T_{latency,FC} = NT_s + \frac{QNM_k + QNK}{IPS} + k\frac{KNQ}{BW}\,. \tag{4.6}$$

Finally for Pipelines 3.9 and 3.11 the schedule is more complex than the straightforward round-robin approach as used in the fully connected case if at least some of the parallelism is to be exploited to lower the latency. Indeed, the communication between nodes of a subbranch can happen simultaneous with the communication between nodes in another subbranch. Following the fusion-diffusion flow of Section 3.3.2, the leaf nodes are the first to transmit their compressed data to their uplinks. This can be done in round-robin fashion: the uplink node receives from each of its leaves one after the other. Using the model developed in Section 4.2 to determine how much words each leaf node needs to send (as given in Tables A.8 and A.10), the time $T_{\to k}$ it takes for all leaf nodes m of their uplink node k to transmit their data to node k is given by:

$$\begin{aligned} T_{\to k} &= \sum_{m \in \mathcal{N}_k \setminus \{n_{uplink}\}} T_{m \to k} && (4.7) \\ &= \frac{1}{BW} \sum_{m \in \mathcal{N}_k \setminus \{n_{uplink}\}} NQ && (4.8) \\ &= \frac{(n_k - 1)NQ}{BW}\,, && (4.9) \end{aligned}$$

with n_{uplink} the uplink of node k and $T_{m \to k}$ the time it takes node m to transmit to node k. Furthermore, all uplink nodes of leaves can communicate with their leaf simultaneous with other uplink of leaves nodes as it was assumed that there are no collisions.

After a (non-leaf) node m has received all compressed observations from its children, it can combine these. From the complexity models it is known that this takes $QN(n_m - 1)$ operations (Tables A.3 and A.5). Therefore the processing delay before a non-leaf node m (different from the root) can transmit to its uplink is:

$$T_{comp,combine,m} = \frac{QN(n_m - 1)}{IPS}\,. \tag{4.10}$$

A similar transmission schedule can then be used on the next levels of the tree. Child nodes of a node that are ready to transmit can be thought to enter a FIFO queue specific to the node. For a child node m, this takes $T_{\to m} + T_{comp,combine,m}$ seconds where $T_{\to m}$ is defined as the time it takes for node m to receive all required signals from its children, counted from the beginning of the inter-node communication (i.e. excluding T_{window} and T_{comp}). Note that this definition of $T_{\to m}$ is a generalization of (4.7) to non-leaf nodes. The first child node in the FIFO queue is allowed to transmit to the uplink node. Child nodes that are not in the first place of the queue need to wait until the node in the first place has finished its transmission.
The advantage of this scheme as compared to the simpler round-robin scheme can be understood by noting that the the duration from the end of the window (which is the same for all nodes as the algorithms in this work are synchronous) to a node being ready to transmit to its uplink is dependant on the number of children it has

as well as on the latency of those children.
For example, a node with many leaf nodes as children may take longer before it is ready to transmit than a node that has just one child which itself has a leaf node as its child. This time difference can be used to hide the latency introduced by the communication of the uplink of those two nodes with the second node. If the time difference is big enough, the second node may even complete transmitting before the first node has received all of its signals as is illustrated in Fig. 4.1. In a round-robin scheme however, the first node would be selected (w.l.o.g.), blocking the uplink from receiving until it has received from that first node, while it could have been receiving from the second node while waiting instead.

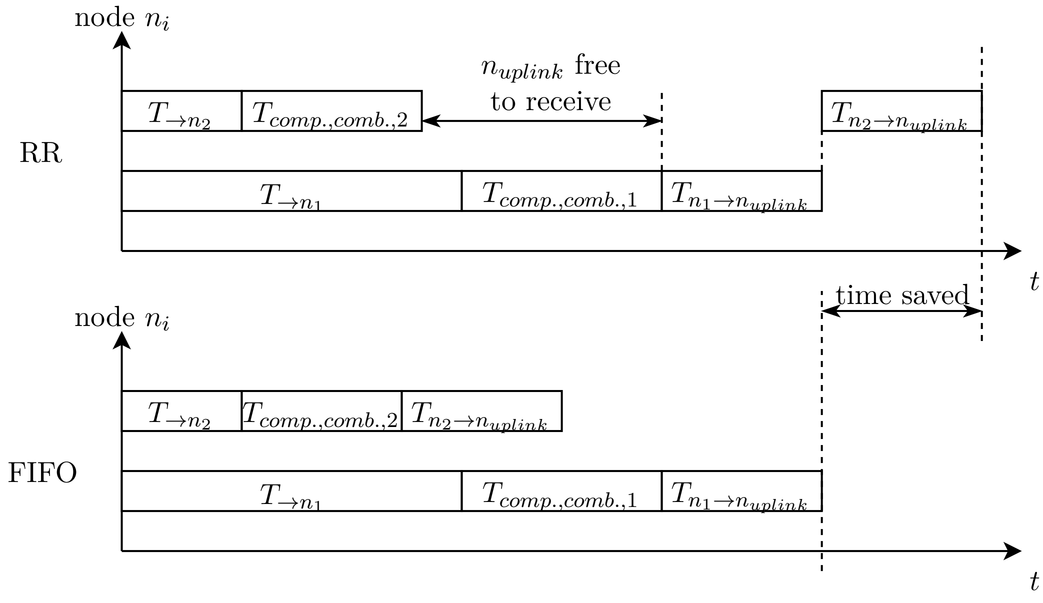


FIGURE 4.1: Example illustrating the differences between round-robin (RR) and FIFO scheduling in the fusion flow communication in tree networks. Relative length of blocks not to scale.

This schedule can then be used recursively on the next levels of the tree until reaching the root node at which the fusion flow has ended and the diffusion flow can be started.

As this scheduling is heavily dependant on the specific structure of the tree topology used, no analytical formula for the latency can be derived as was the case for the other algorithms (cf. supra). The latency will thus have to be determined by simulating the FIFO scheduling for a specific tree topology.

An upper bound for $T_{\to m}$ of a non-leaf node can be seen to be:

$$T_{\to m} \leq \left[\max_{i \in \mathcal{N}_m \setminus \{n_{uplink}\}} \left(T_{\to i} + T_{comp,combine,i}\right)\right] + \sum_{i \in \mathcal{N}_m \setminus \{n_{uplink}\}} T_{i \to m} \tag{4.11}$$

$$= \left[\max_{i \in \mathcal{N}_m \setminus \{n_{uplink}\}} \left(T_{\to i} + T_{comp,combine,i}\right)\right] + \frac{(n_m - 1)NQ}{BW}\,, \tag{4.12}$$

which corresponds to simultaneous receival and combination in the child nodes followed by round-robin transmission to node m once all nodes have finished processing (see Fig. 4.2). By setting the maximum in the equation above to 0 for leaf nodes, it is clear that (4.12) is indeed a generalization of (4.7).

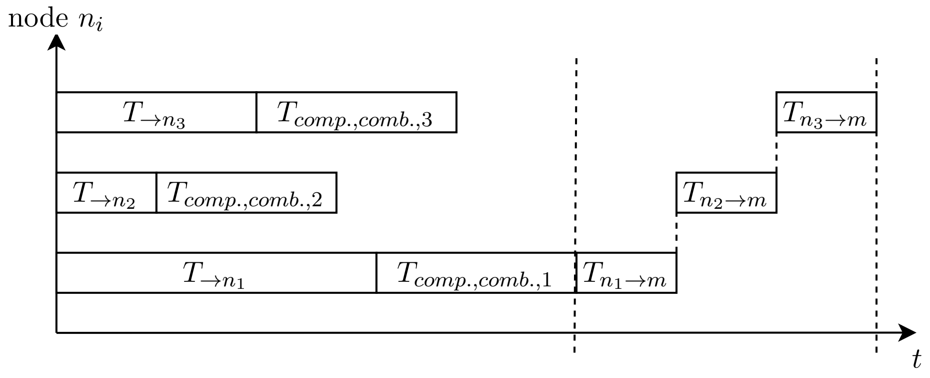


FIGURE 4.2: Illustration of the schedule used to determine an upper bound on $T_{\to m}$ in a tree topology for a non-leaf node m

In the diffusion flow, the root node of the tree combines and sends those signals, which are unique to each child, to each of its children, which then further diffuse that signal to their children (see Section 3.1). For any non-root node k, the time it takes until it has diffused to its children $T_{k\to}$ is given by:

$$T_{k\to} = T_{n_{uplink}\to} + (n_k - 1)T_{comp,combine,k} + \sum_{i \in \mathcal{N}_k \setminus \{n_{uplink}\}} T_{k \to i} \tag{4.13}$$

$$= T_{n_{uplink}\to} + (n_k - 1)\left(T_{comp,combine,k} + \frac{NQ}{BW}\right)\,, \tag{4.14}$$

For a root node r this is

$$T_{r\to} = T_{\to r} + n_r T_{comp,combine,r} + \sum_{i \in \mathcal{N}_r} T_{r \to i} \tag{4.15}$$

$$= T_{\to r} + n_r \left(T_{comp,combine,r} + \frac{NQ}{BW}\right)\,. \tag{4.16}$$

The total latency to do the communication to produce an output is then given by

$$T_{latency,tree} = \max_k \{T_{k\to}\}\,, \tag{4.17}$$

i.e., using the path with the largest latency due to the diffusion flow, which necessarily ends in a node that transmits to leaf nodes.

4.3.2 Update processing delay

After the output is produced, the feature extraction is updated to reflect the most recent set of observations.
The updating consists of the following steps, which are taken right after the output is produced:

Step 5:	Compute and broadcast **L** matrices
Step 6:	Update the covariance matrix of the current class
Step 7:	Solve the compressed GEVD problem for both cases
Step 8:	Resolve the sign ambiguity
Step 9:	Broadcast $\mathbf{G}{-q}$
Step 10:	Compute and broadcast/diffuse $\bar{\mathbf{x}}_q^{i+1}$ (only with re-use)
Step 11:	Update the filter matrix
Step 12:	Update the received compressed observations (only with re-use)
Step 13:	Diffuse the updated received compressed observations (only with re-use in tree topologies)

The same procedure as in the previous subsection can now be used to determine the delay due to each of these steps: the time to do computations is again the number of operations (as given by the models in Section A.1) divided by the processor's instruction rate and the time it takes to do the necessary communication over the network is determined by the number of words needing to be communicated (as given by the models in Section A.2), the bandwidth and the communication schedule. The diffusion parts of the schedules from the previous subsection can be reused for the updating as it requires diffusion flow to update (see also Section 3.3.2 for the discussion of the updating data flow in tree topologies).

To operate according to the pipelines as described in Section 3.6 of the previous chapter, the updating must be finished before the next set of observations is ready for processing. Hence, a complete iteration of the pipeline must be completed in the timespan of 1 window: starting from the end of a window when the observations are all acquired in each node, until the observations of the next window are all acquired. Hence, this upper bounds the allowable total latency, i.e. the sum of the input-output latency and the update processing delay, to be smaller than T_{window}. This is a hard constraint for feasibility.

Observe that, as will be discussed in the following chapter, this upper bound is not easily increased by enlarging T_{window}. Indeed, this would proportionally increase the time needed for computation and communication hence proportionally increasing the latency as well. A simple solution that is effective, is to only update periodically rather than in each window. Hence the update processing delay can be amortized over multiple windows. Note that this cannot be done for the steps producing the output or else the BCI would be essentially disabled during all windows except the one after which the updating occurs.

4.4 Energy model

The energy consumption E of a node mainly consists of two components, which are both influenced by the choice of algorithm: energy consumed for computation E_{comp} and energy consumed to send and receive data via the wireless transceiver E_{comm}. The latter can be split up further into energy consumed to transmit data E_{tx} and energy consumed to receive data E_{rx}:

$$E = E_{comp} + E_{comm} = E_{comp} + E_{tx} + E_{rx}\,. \tag{4.18}$$

It is assumed that each node is of the same design. Therefore, the node that needs the most energy to run the pipeline will deplete its battery first and hence limits the battery life of the system. Therefore, the energy consumption can equivalently be studied by only studying that node (which still requires estimating the energy consumption of all nodes to identify which node that is, but simplifies presentation of the results).

Computation-related energy

The computation-related energy consumption E_{comp} can be estimated by combining the estimate for the number of operations N_{comp} given by the computational complexity model, with an estimate of the computational efficiency of the processor η [J/FLOP]:

$$E_{comp} = \eta \cdot N_{comp}\,. \tag{4.19}$$

Note that here indeed it makes sense to use the average number of operations, as discussed in Section 4.1.

Communication-related energy

To estimate the communication-related energy consumption E_{comm}, the communication bandwidth models of the individual algorithm steps are combined with a propagation model that specifies how much energy is needed to transmit data over a certain distance.

As the sensor nodes are to be placed on the wearer's head in some fashion, there is no line of sight between sensor nodes. According to Zasowski et al. [50], the contribution of the direct path between nodes is negligible due to the strong attenuation by the head. Rather, the communication is established by creeping waves around the contours of the head. As such the first order free space model wherein energy is quadratically related to the distance, is not valid for non-line of sight communication

(NLOS) around the human body. Instead the model needs to be generalized to use an exponent $n = 5 - 6$ for NLOS communication (instead of $n = 2$ as for communication in free space) [51, 44]:

$$E_{tx} = E_{TXelec} \cdot k + E_{amp} \cdot k \cdot d^n \,, \tag{4.20}$$

$$E_{rx} = E_{RXelec} \cdot k \,, \tag{4.21}$$

with E_{TXelec}, E_{TXelec} respectively the energy the wireless stack dissipates to run the circuitry for the transmitter and receiver, E_{amp} the energy for the transmit amplifier and k the number of bits sent.

From the measurements in [52] can be deduced that this model also holds for communication by nodes around the head, where d should be interpreted as the path length along the contour of the head between two nodes, rather than the length of the direct path.

For a wordsize of S bits, the model can be rewritten as

$$E_{tx} = (E_{TXelec} \cdot + E_{amp} \cdot d^n) \cdot S \cdot N_{comm} \,, \tag{4.22}$$

$$E_{rx} = E_{RXelec} \cdot S \cdot N_{comm} \,, \tag{4.23}$$

with N_{comm} the number of words sent in that transmission.

Using N_{comm} as given for each communication by the communication bandwidth models from Section 4.2 and d calculated from the nodes positions, the energy needed to do each of the communications can now be estimated.

Chapter 5

Evaluation

This chapter presents the evaluation of the pipelines that have been proposed in Chapter 3. The goal is to investigate whether the proposed distributed signal processing algorithms, integrated in a end-to-end classification pipeline, can make a WESN a viable approach to implementing a wearable EEG-based BCI system, suitable for long term use, and comparing the proposed approaches' performance to one another.

The evaluation will examine and compare the performance based on the three objectives that were presented in Section 1.4 as well as look at feasibility w.r.t. constraints of the WESN and hardware. To do so, the models for the latency and the energy consumption presented in the previous section will be used, together with an estimate of the decoding accuracy by emulating a WESN on experimentally recorded data.

This experimental setup and the emulation of the WESN will be detailed in Section 5.1. Next follow four sections investigating the influence of respectively the network size, (efficient) re-use as discussed in Section 3.5, the amount of electrodes per node and finally the influence of technological parameters. This chapter than concludes with a discussion of the results from these four experiments.

> Note that the topic of this evaluation is to investigate the performance characteristics of the distributed systems rather than achieving the highest possible accuracy. Therefore, the accuracy can potentially be further increased by per-subject hyperparameter tuning. This is however outside the scope of this text and should be the topic of future work.

5.1 Experimental setup

As no physical realizations of the envisioned WESN (nodes) exist, *high density EEG (HD-EEG)* recordings will be used to mimic recordings from a WESN as envisioned

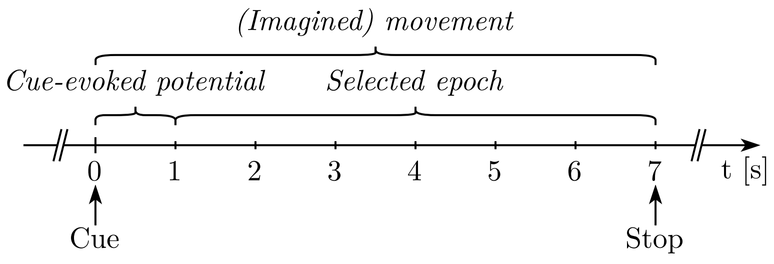


FIGURE 5.1: Experimental protocol timing

in this text.

The reason HD-EEG is a suitable platform for mimicking and consequently studying WESNs is twofold.
First, due to the high density of the electrodes, the distance between neighbouring electrodes in HD-EEG is smaller than that of the envisioned nodes. While this does restrict the flexibility to miniaturize the node (and thus electrode distances) further, doing so would decrease the extra information due to spatial effects [53] and is thus undesirable. The data from a node in a certain location on the scalp is hence comparable between HD-EEG and an envisioned node.
Furthermore, as all channels are recorded without compression, HD-EEG recordings also allow to simulate for example different topologies, algorithms and parameter values post factum instead of requiring multiple experiments.

5.1.1 Experimental protocol

In [53], 255-channel HD-EEG data was collected for multiple tasks, including MI. 30 able, male subjects between 22 and 35 years old participated in the experiment. The EEG was recorded using a SynAmps RT device (Compumedics, Australia) with active Ag/Cl electrodes at a sampling rate of 1 kHz. The electrodes were placed on the head according to the international 10-5 (5%) system. The `FCz` electrode was used as an active reference and `AFz` as the ground.

Each subject performed 50 trials with imagined movement. The experimental protocol for each trial was as follows: At $t = 0$ s, a visual cue was displayed on a display in front of the subject with text that prompted the subject to perform (imagined) movement with either their left or right hand. After 7 s, the subject was asked to stop the (imagined) movement. Between the trials there was enough time as to not influence each other. This is illustrated in Figure 5.1. Hence the recordings are the result of a synchronous, reactive system.

5.1.2 WESN emulation

Next a WESN topology is emulated based on the EEG electrode locations. To do so, the pre-processing as explained in Section 2.1 is first applied per-channel. Next,

Model	Parameter	Value		Source
General	S	32	bits	–
Energy	η	0.1	nJ/flop	[54]
	E_{TXelec}	16.7	nJ	[44]
	E_{RXelec}	36.1	nJ	[44]
	E_{amp}	7.99	µJ/m	[44]
Latency	IPS	1.0	Gflops/s	[54]
	BW	1.0	Mbits/s	[55]

Table 5.1: Hardware specifications used in simulations

nodes are constructed by first selecting for each node, a set of electrode locations from those available in the HD-EEG electrode cap. From each node's set, one electrode is selected to be the local reference for that node. The recorded data of the channels corresponding to each of the nodes' satellite electrodes are then re-referenced to the corresponding node's reference electrode. It is assumed that the processing and communication system of the node is located at the reference electrode. Further note that as these nodes are envisioned to be stick-on patches or otherwise very flat and close to the skin, the areas covered by the electrodes of different nodes cannot overlap.

To also be able to evaluate the tree topology algorithms, Dijkstra's shortest path algorithm (see [45]) was used to prune the topology presented above into a tree topology. The metric that was used as the cost of a link is the arc distance on a least-squares fitted spherical model of the electrode locations raised to the power n as this represents the part of the communication energy that is influenced by the distance (see Section 4.4).

Finally the emulated WESN data is then fed to the distributed classification pipelines of Section 3.6 in order to simulate the complete system's behaviour and assess its performance.

The models of the energy and latency will also be applied to the emulated topology from this section. As there is no physical implementation of the envisioned nodes, the hardware specifications needed for the following models are taken from the relevant literature of each component. These specifications are summarized in Table 5.1.

5.1.3 Training and testing

In order to initialize a pipeline, the first 34 trials were concatenated and used as to determine a long-term estimate of the covariance matrices, which were then subsequently used to initialize the filters using the classic CSP method and finally, the resulting outputs from those initial filters were used to train the classifier. This initialization is done specific per subject.

Next, the pipeline is emulated on those same 34 training trials in order to initialize the compressed covariance matrices and filters. The emulation is then allowed to continue running on the remaining 16 trials, on which the decoding accuracy is measured[1].

In practice, this initialization procedure could be accomplished by first having the subject perform a cued recording session, similar to how the HD-EEG experiment was performed. Afterwards, the system can then be programmed with the initialization based on the results of that first session and continue to further adapt while in use. Ideally, this recording session should be eliminated, e.g. using subject-independent initialization or by having the procedure built into the final system.

5.2 Experiment 1: Influence of network size and node placement

While this thesis does not focus on the aspects of miniaturization w.r.t. accuracy (as for example in [53]), it is prudent to briefly investigate how the choice of topology influences the performance. This will allow to study and compare other properties of the different proposed pipelines to each other in a realistic setting. Note that scaling behaviour w.r.t. certain parameters is not necessarily of interest for the current evaluation: this behaviour is only valid at extremes whereas the optimum could potentially be more moderate.

In this section, the size of the network as well as the rough placement of the nodes will be studied together in a single experiment.

5.2.1 Methods

Using the experimental setup from Section 5.1, 8 WESNs with different topologies were emulated. These are shown in Figure 5.2. They are numbered in order of growing size except for topologies no. 3 and 4 which only differ in the placement of the non-central nodes. Following the symmetry of the sensorimotor areas, all of the topologies were chosen to be symmetrical. The electrodes of each node were placed to give a combination of multiple directions to maximize performance [16]. When selecting the satellite electrodes, a distance to the reference electrode $>3\,\text{cm}$ was aimed for to ensure performance is not degraded[17]. The amount of electrodes was determined using manual exploration. For each of the topologies, all 3 of the pipelines were then emulated and all other parameters were set according to Table 4.1 and the technical parameters for the models according to Table 5.1.

[1]Note that the pipelines are still given a label to correctly update its covariance matrices (see Section 2.2), hence introducing bias.

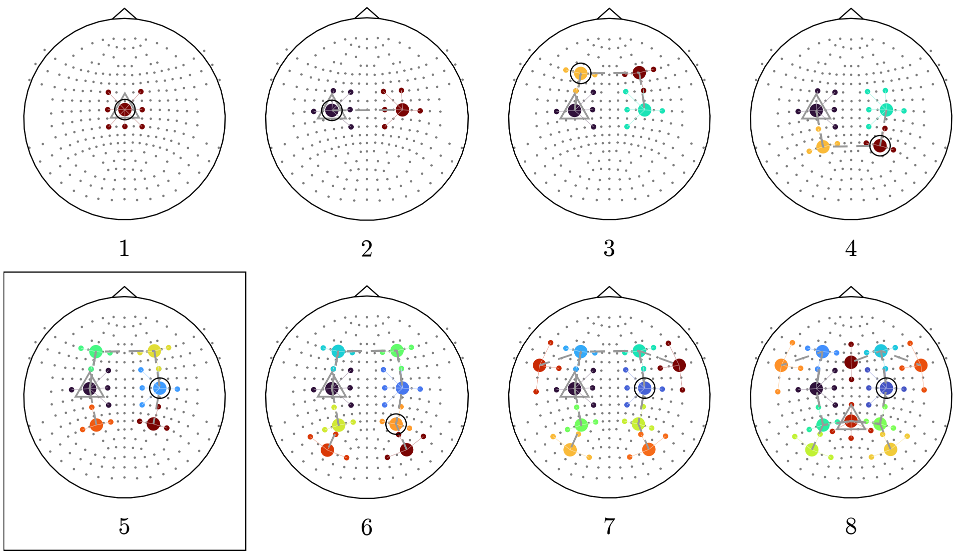


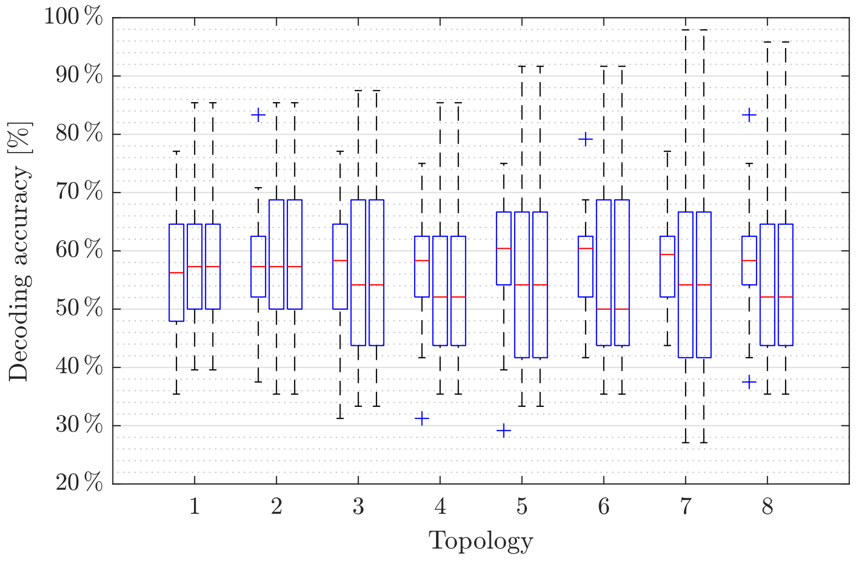
FIGURE 5.2: Topological maps of the scalp showing the topologies used for the evaluation. The topology outlined with a rectangle will be used as the default topology (see Section 5.2). The locations of the electrodes in the HD-EEG cap are indicated with grey dots. The electrodes (dots) and connections (solid lines) of each node are indicated with a different colour. A small(large) dot represents a satellite(reference) electrode of the node with that colour. Dashed grey lines indicate inter-node connections of the pruned tree, while a grey triangle indicates the tree's root.

5.2.2 Results

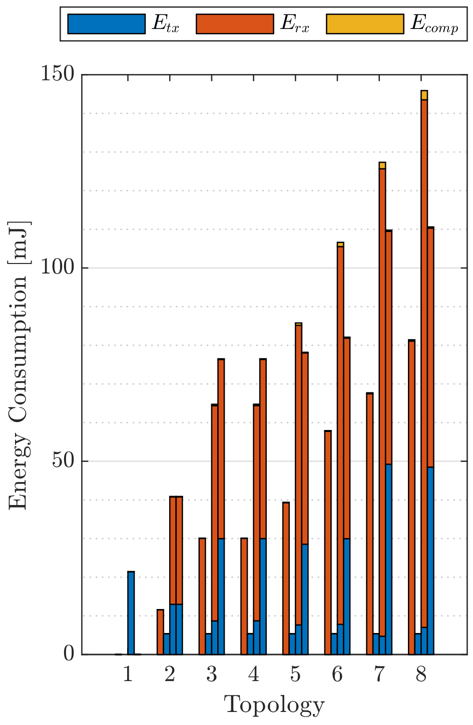
The results of this experiment are shown in Figure 5.3. The different pipelines are grouped per topology. The centralized pipeline on the left, the fully connected pipeline (with re-use) in the middle and finally the tree pipeline (with re-use) on the right. In Figure 5.3b, the first bar of each group represents the energy consumption by the node acting as the FCe. An extra bar has been inserted in the second position, representing the mean energy consumed by the other nodes. As the FCe dominates the latency, no such extra bar is needed there.

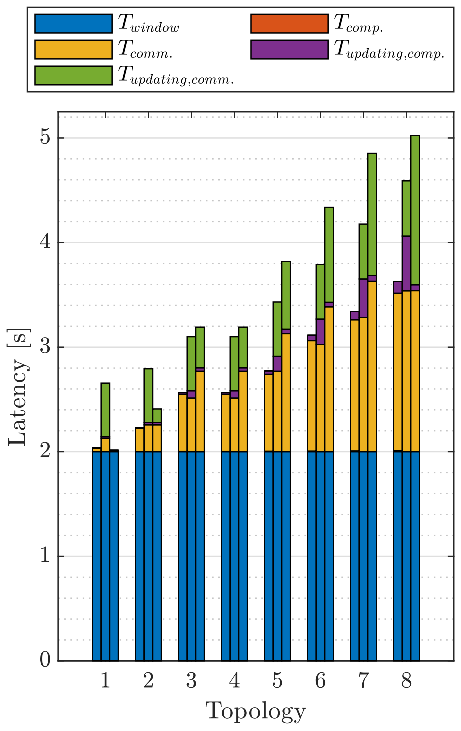
While the decoding accuracy varies significantly over the different subjects, due to the fixed layout imposed by the electrode cap, the energy and latency are assumed to be identical for each of the subjects. Subsequently, they can be visualized using a stacked bar graph, allowing to visualize the individual components contributing to the sum. These correspond to the components identified in Section 4.4 and 4.3 for respectively the energy (Subfigure b) and latency (Subfigure c).

Looking at the accuracy of the centralized pipeline in Figure 5.3a, it is clear that the accuracy first increases with the growing topology size, reaches a maximum and then decreases. Choosing a small topology might thus be practical for the envisioned use



(A) Decoding accuracy for the centralized pipeline (Alg. 3.1) for the topologies of Figure (5.2)





(B) Energy consumption for the centralized pipeline (Alg. 3.1) for the topologies of Figure (5.2)

(C) Latency for the centralized pipeline (Alg. 3.1) for the topologies of Figure (5.2)

FIGURE 5.3: Influence of network size and node placement on performance of each of the pipelines. The groups are as described in Section 5.2.2.

case, but it reduces the amount of information that can be extracted and as such, reduces the decoding accuracy. On the other hand, choosing a too large topology also decreases the decoding accuracy. Among others, this can potentially be caused by the increasingly low amount of extra information adding extra channels adds while it does add noise. Also, as the network size and thus the size of the GEVD problem grow, the problem becomes more prone to overfitting. In this case, the best performing topologies are no. 5 and 6 which perform similarly in terms of their median decoding accuracy.

However, when looking at the other two metrics, energy and latency, it becomes clear that the smallest topology that gives satisfactory decoding results is to be preferred. Indeed, as the size of the WESN grows, both the necessary time for computations as well as for communication increase. At the same time, the energy consumption of both also increases.
The former follows from the increased size of the matrices involved in the calculations. As shown in Section 4.1, this causes the necessary number of computations to increase as well, increasing the required computation time and hence also energy.
The latter, in the centralized and fully connected pipelines and in accordance with the models of Section 4.3, is caused by an increasing amount of channels that need to be received from other nodes, increasing both the time and energy required to do. As the amount of data that needs to be transmitted by each node is independent from the amount of nodes, only the receiving is affected.
For the tree topology however, there are two mechanisms at work. First, each neighbour gets transmitted a different signal and as with an increase in amount of nodes, the average number of neighbours is expected to increase, both energy and latency increase, now due to both received as well as transmitted signals[2]. The reduced energy consumption due to the shorter distances in the tree network was hence completely negated. Second, the depth of the tree increases as well. Therefore, it takes longer for a signal to diffuse to the leaves and fuse back to the root, increasing the latency even more. Note that topologies no. 3 and 4 have the same latency. Looking at their construction in Figure 5.2, this is expected as they each have the same number of nodes.

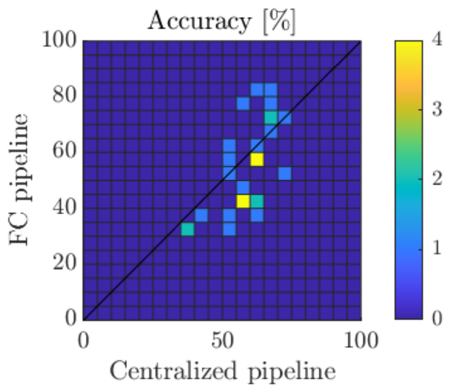
In this experiment, topology no. 5 would hence be preferred as this yields the same, highest accuracy as topology no. 6 but with lower latency and energy consumption. Looking at the node locations of topology no. 5, they can be seen to cover the extended region (both anterior and posterior) around the sensorimotor areas, which is in accordance with physiological theory.

Comparing the different pipelines to each other now, it can be seen that the median accuracies of the distributed pipelines are lower that that of the centralized pipeline,

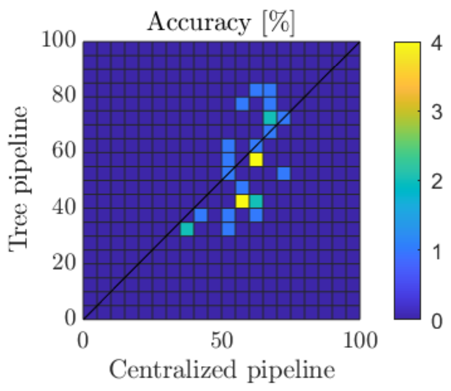
[2]There exists also a method where each node broadcast a single compressed set of signals to its neighbours rather than a different set per neighbour [48]. As it is more complicated, it was not considered here.

except for the smallest topology. Furthermore, the standard deviations of the distributed pipelines are, safe for the smallest topology, larger than that of the centralized topology.

However, it is still unclear what causes this standard deviation to be enlarged: are good performing subjects improving and vice versa for poor performing subjects, or do they seemingly improve or degrade randomly? Figure 5.4 shows heatmaps of a 2D-histogram of the centralized pipeline's decoding accuracy of each subject on the x-axis vs. those of respectively the fully connected and tree pipelines on the y-axis. Although subtle, the lower performing subjects in the centralized pipeline tend to stay below the black line indicating the points with equal accuracy in both pipelines. Their accuracy thus lowers going to the distributed pipelines. On the other hand, the opposite can be seen for the subjects with a higher centralized accuracy.



(A)



(B)

FIGURE 5.4: Heatmaps linking the per-subject accuracy for a centralized and a distributed pipeline

When comparing the energy and latency of the different pipelines to one another, the centralized pipeline, surprisingly, outperforms the distributed pipelines. It is also the only pipeline that remained feasible for all topology sizes. The reason for this is the small topology sizes, both in terms of amount of nodes as well as electrodes. As the distributed algorithms add overhead (a.o. transmitting filter updating information back to the rest of the network and waiting for every node to have received all necessary data, both clearly increasing the corresponding components in Figures 5.3b and 5.3c as well as re-use, which will be discussed in more detail in Section 5.3), the benefits of using a distributed pipeline do not outweigh the disadvantage due to the overhead for topologies of this size.

Indeed, when repeating the experiment to determine the energy and latency for a large topology of a few hundred nodes[3], the distributed pipelines' energy and latency are indeed lower, although it is unclear how this would affect convergence and thus accuracy. Especially so for the tree pipeline, the energy and latency are significantly lower than those of the fully connected pipeline in such a large network due to a divide-and-conquer-like mechanism avoiding a single node having to be connected to every other node. This reduces both energy consumption and latency and justifies the extra overhead. For the energy consumption, due to the low average number of neighbours in the pruned tree networks, the energy already becomes lower for the tree topology than for the fully connected one. However, again because of the low average amount of neighbours, the diffusion-fusion flow increases the latency beyond that of the fully connected pipeline.

5.2.3 Conclusions

In summary, to minimize energy and latency while maximizing decoding accuracy, the smallest topology that gives satisfactory accuracy results is to be preferred. Furthermore, increasing the topology size yields diminishing returns w.r.t. accuracy or could even lower it. The best topology is hence no. 5 and will be used as the basis for further experiments in this work. As the topology is of relatively small size, the more lightweight centralized pipeline here outperforms the distributed ones. All pipelines are however still feasible w.r.t. the latency for this topology.

The distributed pipelines lend themselves well to topologies with a high amount of nodes. Depending on whether energy or latency is to be optimized, respectively the tree or fully connected pipeline should be chosen.

5.3 Experiment 2: Impact of (efficient) re-use

In Section 3.5 it was discussed that it might be necessary to increase the convergence rate of the distributed algorithms to keep up with the rate at which the statistics change. Although more iterations per window can be performed by repeating the steps on the same window of data to boost the convergence rate, it was explained that this incurs a multiple amount of energy consumption as well as latency compared to a single iteration per window. This led to the development of a more efficient approach to re-using data in [47] for fully connected topologies, which was extended to tree topologies and integrated in the pipeline in Section 3.5.

This section now evaluates whether the accuracy indeed improves when employing re-use, and what the impact is of doing this efficiently or non-efficiently, on the energy consumption and latency.

[3]As such large WESNs are not relevant for the application considered in this thesis, they will not be discussed in depth but will rather be used to complement the results in order to demonstrate interesting aspects of the evaluated approaches which only occur for larger networks and as such, might be valuable to other applications where larger networks are used.

5.3.1 Methods

To investigate this, the base topology from the previous section together with the typical parameters given in Table 4.1 and the technological parameters given in Table 5.1 is simulated using the setup as described in Section 5.1 for various amounts of re-use cycles R ranging from 1 (no-reuse) to 10.

5.3.2 Results

The results are shown in Figure 5.5. From Subfigure a it is clear that re-use does not improve the median decoding accuracy for neither distributed topology for this network (compare $R = 1$ with $R > 1$). The standard deviation is also lower in the centralized pipeline than in the others. As was the case in the previous section, repeating the experiment with a network with a few hundred nodes, did show improvement when using the more elaborate technique. Hence, it is hypothesized that because of the small size of the network, the problem can converge sufficiently quick, even with just 1 iteration per window, whereas when the network and thus the problem grows in size, also the number of variables that needs to be optimized grows and hence a single iteration is no longer sufficient. For the application considered in this text however, the useful topology sizes are too small to reap benefits from re-use.

In both the energy consumption and latency figures, the centralized pipeline outperforms the distributed ones due to the overhead as discussed in the previous section. Again, as also discussed in the previous section, when repeating the experiment with a network with a few hundred nodes, the distributed pipelines outperform the centralized one for larger networks.

As applications that use such large topologies can still benefit from re-use, it is still useful to investigate the impact of efficient re-use as compared to non-efficient re-use for the current experiment. In Figures 5.5b and 5.5c, the total energy respectively total latency with non-efficient re-use is indicated with a solid line for both fully connected and tree topologies. While both efficient and non-efficient linearly increase with R, the increase is less steep for efficient re-use than for non-efficient re-use showing that, although the energy consumption and latency will continue to increase for both, efficient re-use will outperform non-efficient re-use for any value of R.

Observe that the energy consumption of the tree topology eventually becomes bigger than that of the fully connected topology. Indeed, while E_{rx} and E_{comp} increase at a similar rate, E_{tx} increases slower for the fully connected topology than for the tree topology. This can also be seen in the models of Table 4.3: N_{tx} scales only with $\frac{R}{K}$ for the fully connected topology while it scales with R for the tree topology. The reason for this is that in the tree topology, every re-use cycle requires each node to participate in a diffusion cycle with the updating node as the root (see Section 3.5.2), while in the fully connected network, the updating node need only broadcast and all other nodes only need to receive (hence the same E_{rx} scaling for both). As the communication-related energy is on average linear in the number of words, this explains the observed effect.

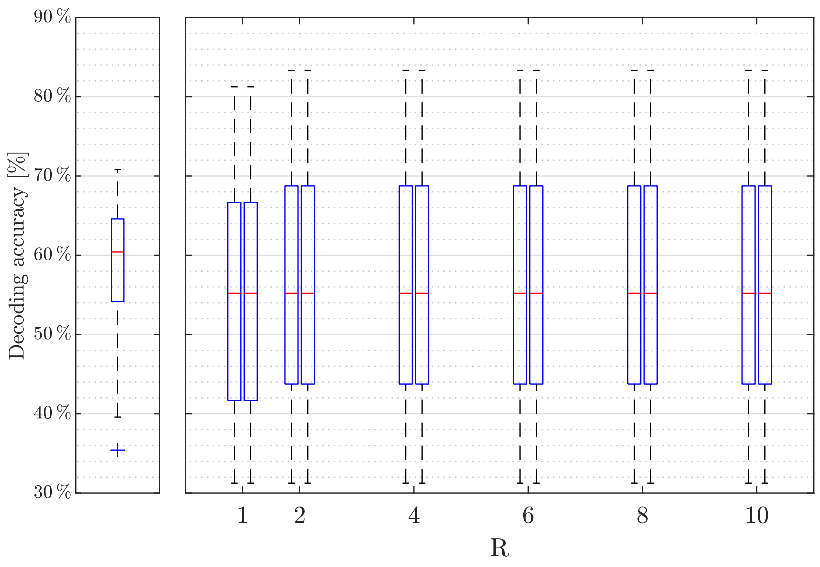
Concerning the latency of the distributed pipelines, the latency due to updating-related computations can be seen to increase at a similar rate in both pipelines. The latency due to the updating-related communication however increases at a much slower rate for the tree pipeline.

5.3.3 Conclusions

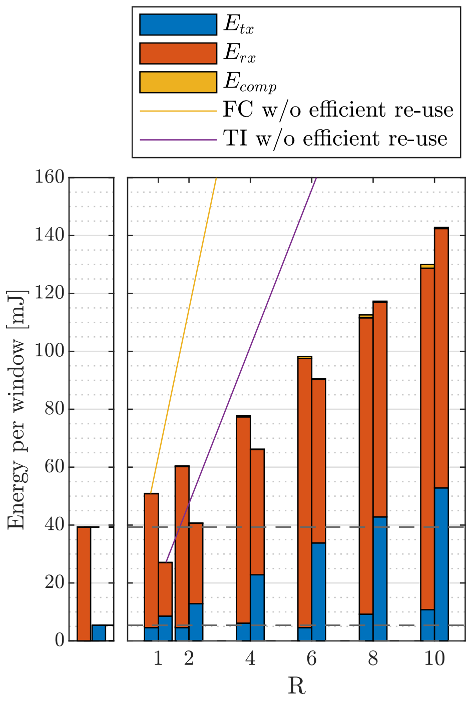
For the systems considered in this thesis, re-use does not improve the median accuracy. The centralized pipeline is still more accurate in decoding the correct output. Also w.r.t. energy consumption and latency, the centralized pipeline outperforms the distributed pipeline.

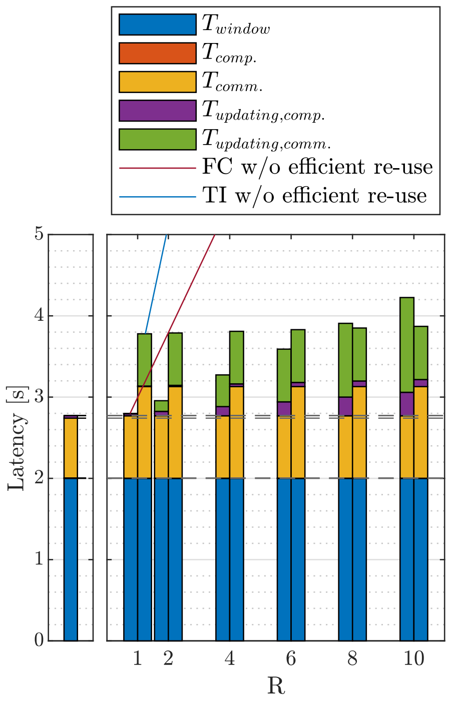
However, for other applications with larger networks were distributed pipelines outperform the centralized one (see Section 5.2), the advantage of efficient re-use as compared to non-efficient re-use is significant and continues to increase with R. Furthermore, although a tree topology becomes more efficient compared to a fully connected topology as the network grows in size, its energy increases more rapidly with R than that of the fully connected topology, since it has to participate in a diffusion flow in every re-use cycle. For the latency, the opposite is true.

Note that for larger networks, it might also be possible that there exists a balance between the size of the network and R. Indeed, a smaller network requires less energy and incurs a lower latency per iteration which can then subsequently be used to increase R (see Section 5.2). The maximum achievable accuracy decreases (i.e. that of the centralized pipeline as the distributed algorithms converge to the same solution), but as more iterations can be done towards convergence, the accuracy might be better for the same energy consumption or latency. However, as this is only applicable to larger networks where re-use affects the performance, investigating this effect is outside the scope of this text.



(A) Decoding accuracy





(B) Node energy consumption per window of data

(C) Latency

FIGURE 5.5: Influence of (efficient) re-use on performance of each of the pipelines. The groups are as described in Section 5.2.2.

5.4 Experiment 3: Influence of the per-node channel count

For the design of a hardware implementation, it is useful to know what the impact of the number of channels per node is on the performance of the pipelines. This has several reasons:

- A smaller amount of channels makes the node inherently smaller and cheaper to both design and manufacture.
- The number of channels might impact performance.
- If the energy consumption and latency are affected, there might be a maximum number of nodes that can ever be used to have a certain battery life or to remain feasible w.r.t. the time spent processing.
- While satellite electrodes can always be turned off, adding them afterwards requires a redesign. Possibly the optimal amount of channels is also subject specific. One could then design a single node with a channel count sufficiently high enough such that electrodes can be selected afterwards, individually for each subject.

5.4.1 Methods

To study the impact of the per-node channel count, the base topology (topology no. 5 of Figure 5.2) is modified such that the reference electrode is kept and an increasing amount of satellite electrodes is then added, as shown in Figure 5.6. The setup of Section 5.1 was then used with otherwise the parameters of Tables 4.1 and 5.1.

5.4.2 Results

Figure 5.7 shows the results of this experiment. For the centralized topology, as the number of channels in each node M_k is increased, also the number of channels that need to be send to the FCe grows, incurring energy consumption per extra channel that needs to be sent.
In the FCe itself, this translates to increasing amounts of energy being consumed to receive the signals as well as increasing amounts of latency to receive all those signals one after the other. From Figure 5.7 this can be seen to be a major factor in the performance w.r.t. energy consumption and latency.
Furthermore, the centralized problem that subsequently needs to be solved grows with the total number of channels in the network M. In this specific case, $M = KM_k$ since M_k was taken to be equal for all nodes $k \in \mathcal{K}$. Hence with $K = 6$ the dimensions of the problem grow 6 times as fast as the number of channels per node. From the model of Table 4.2 however, it is known that the amount of operations to solve the centralized problem increases with the second and third power of M. Consequently, the computation time and required energy rise sharply with M_k and become non-negligible.

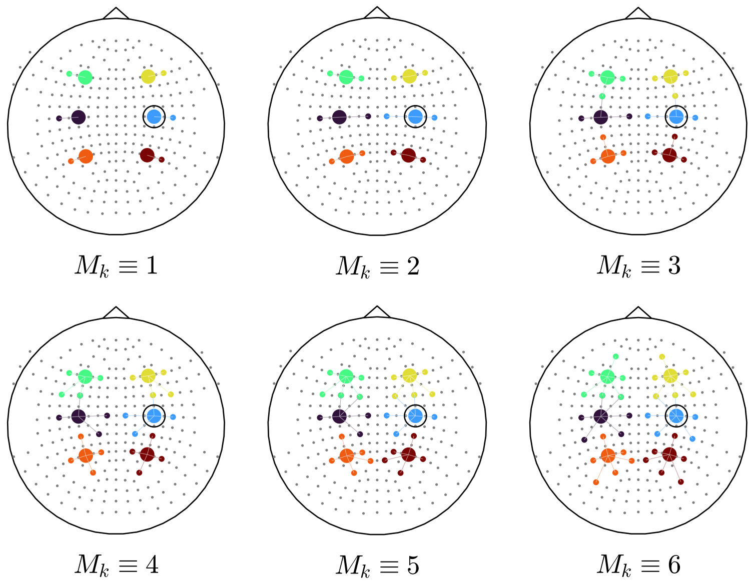


FIGURE 5.6: Base topology modified to have an increasing amount of channels M_k in each node. The reference electrode locations of the base topology (topology no. 5 of Figure 5.2, see Section 5.2.3) are retained. The format of the topoplots is the same as described in Figure 5.2.

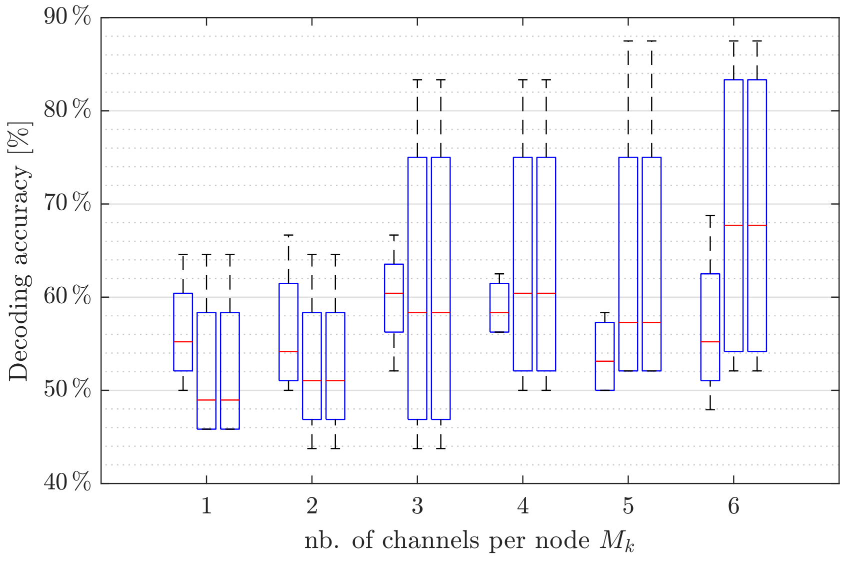
On the other hand, as can be seen from the figure, both energy consumption as well as latency increase only marginally. This is also clear from the models given in Tables 4.3 and 4.2: the signals that a node needs to send or receive is independent from the channel count of its neighbours as due to the compression, each node need only send Q signals to each neighbour (using broadcasting in the fully connected case). The biggest factor lowering performance in the centralized pipeline, increased communication, is thus not present here. The other factor, the number of computations, is present, but with a different scaling behaviour. Indeed, from Table 4.2 it can be seen that the necessary amount of computations increases only quadratically, rather than cubically. Consequently the increase in latency and energy consumption is negligible.

Figure 5.7a, which shows the accuracy for each topology with a certain M_k, motivates why to have multiple channels per node in the first place. Indeed, as M_k increases from a single channel, the median performance increases as well. This trend drops off after 3 channels but surprisingly, it continues for the distributed topologies with a spread around the median than covers both descent as well as excellent performance. This might also indicate that indeed electrode placement is subject specific.

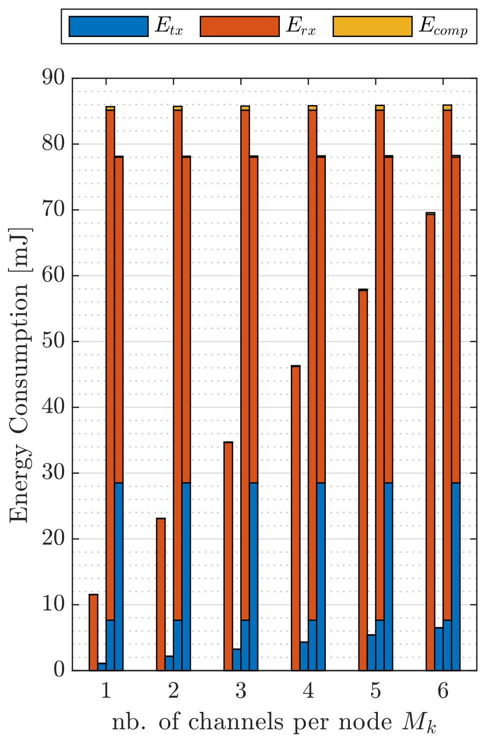
5.4.3 Conclusions

From the results it is clear that there is a bifurcation point: for smaller nodes the centralized pipeline outperforms the distributed pipelines for any of the metrics. Once past a certain number of channels per node (the specific position depends on which metric is considered for the boundary), the distributed pipelines however are more performant and, at least for the latency and energy consumption, demonstrate a more manageable decrease in performance as the number of channels is increased even further.

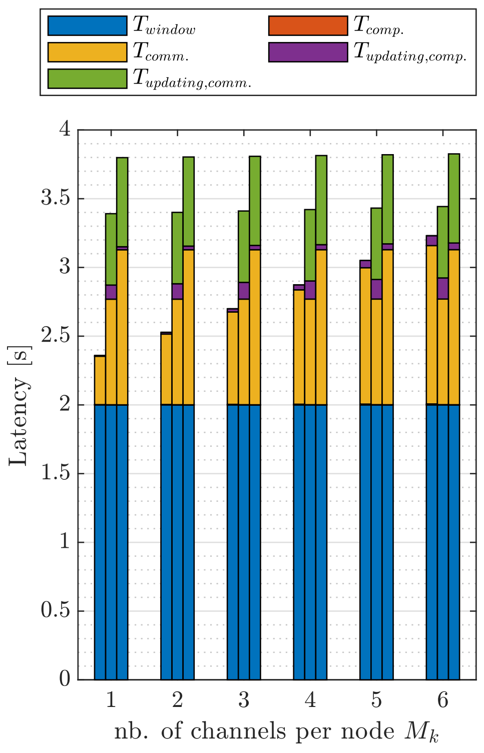
Eventually however, either the battery life becomes unsatisfactory or the system is no longer feasible due to the latency. There is thus a maximum number of channels that can simultaneously be used. If more electrodes are placed on the node, channel selection must be used to reduce the amount to this maximum number, providing the possibility to virtually tune the electrode placement to each user.



(A) Decoding accuracy



(B) Node energy consumption per window of data



(C) Latency

FIGURE 5.7: Impact of the per-node channel count M_k on the energy consumption and latency of each of the pipelines. The groups are as described in Section 5.2.2.

5.5 Experiment 4: Influence of technological parameters

Another important consideration when designing a hardware implementation, is how the performance is influenced by changing the technological parameters concerning the hardware for both processing and communication. For example, when designing for a limited footprint, how should the sizes of the processing and communication system compare if their area is related to their energy efficiency and/or speed? Or can a cheaper component be used that is slower while still remaining feasible?

Until now, the parameters of Table 5.1 have been used. In this section, the influence of these choices will be studied.

5.5.1 Methods

To investigate the influence each of the technological parameters has on the latency and energy consumption, simple models will be fitted based on the more extensive model of Chapter 4. As the technological parameters discussed here do not directly affect the accuracy only the latency and energy consumption will be discussed. Note however that they do influence the accuracy indirectly. For example, lowering the latency for a set of network parameters may then allow to increase the latency again, without becoming infeasible, by tuning the network parameters to have a higher accuracy at the maximally allowable latency.

Energy consumption

For the energy consumption, because there are multiple parameters governing the communication-related energy (see Section 4.4), a model was fitted that scales all communication-related technical parameters with scalar λ_{comm} and the computation-related technical parameter η with the scalar λ_{comm}. As can be seen from the more detailed models of Section 4.4, all the scaled technical parameters relate to the energy linearly. Therefore, the following linear model seems appropriate:

$$\begin{aligned} E \approx\ & a' \cdot \eta \cdot \lambda_{comp} + \Big(b'_{TXelec} \cdot E_{TXelec} \\ & + b'_{amp} \cdot E_{amp} \cdot d^n + b'_{RXelec} \cdot E_{RXelec}\Big) \cdot k \cdot \lambda_{comm} \qquad (5.1) \\ =\ & a \cdot \lambda_{comp} + b \cdot \lambda_{comm} + c\,. \qquad (5.2) \end{aligned}$$

Latency

For the latency, there are only two technical parameters: BW and IPS. They can be seen in Section 4.3 to be inversely proportional to the latency. Hence the following model seems appropriate:

$$Latency \approx \frac{a}{BW} + \frac{b}{IPS} + c\,. \qquad (5.3)$$

Observe that this model is still linear in its parameters a, b and c.

These simplified models for the energy consumption and the latency will be fitted with a least-squares objective. To generate the data to fit to, the detailed models of Chapter 4 are used with the other parameters set as in Tables 4.1 and 5.1 and for the base topology as discussed in Section 5.2.3. The energy models were run with the scaling parameters ranging logarithmically from 1×10^{-4} in 100 steps to 1×10^{1}. The latency models were run with BW and IPS ranging logarithmically from 1×10^{5} to 1×10^{10} in 100 steps. The fitted parameters will then provide insight into the influence a change in one of the technological parameters has on the performance.

5.5.2 Results

The fitted parameters are shown in Table 5.2 for the energy consumption and in Table 5.3. Both tables also show the goodness-of-fit of each model, expressed in terms of the RMSE. All of the models can be seen to indeed closely match the detailed models' results.

Energy consumption

Observe that for the energy consumption, the intercept c is almost zero. Indeed, if it were free to both compute and communicate, the energy consumption would be zero as well as these were the only two sources of energy consumption that were captured in the detailed models and are both present in the simple models. Also note that the computation-related energy has no impact on the energy usage on the nodes other than the FCe in the centralized pipeline as a is close to zero. Indeed, as specified in Algorithm 3.1, these nodes don't have any computations to perform.

In each of the pipelines, $a < b$, indicating that changing the computation-related energy has less of an impact on the total energy consumption than changing the communication-related energy. Hence, the latter should be improved first rather than the former if the systems needs to be made more energy efficient (without changing other system parameters). If the system can use more energy (e.g. such that cheaper components can be used), changing η will have the least impact on the energy consumption for the same decrease of the parameter. Note however that the computation- and communication related parameters might of course scale differently from a hardware perspective; the results of this experiment hence are only half of the technical parameter scaling problem.

Latency

Concerning the latency, the impact of BW and IPS is in the same order of magnitude for each of the topologies. As illustrated in Figure 5.8, BW and IPS can be exchanged, depending on which hardware component is easier to optimize. Notice however that for both the centralized as well as the fully connected pipelines, $a < b$,

	Coefficients			
Pipeline	a	b	c	RMSE
Centralized – FCe	1.0267×10^{-4}	3.9277×10^{-2}	-1.3211×10^{-18}	0
Centralized – others	1.8128×10^{-19}	5.3932×10^{-3}	2.4771×10^{-19}	0
Fully Connected	6.9636×10^{-4}	8.518×10^{-2}	7.3671×10^{-6}	8.0813×10^{-5}
Tree	2.0983×10^{-4}	7.8013×10^{-2}	3.4457×10^{-6}	2.9226×10^{-5}

Table 5.2: Resulting coefficients and RMSE from fitting the model given by (5.2) to the total energy consumption simulation results using the models derived in Section 4.4 for the various pipelines

	Coefficients			
Pipeline	a	b	c	RMSE
Centralized	1.47×10^{6}	1.82×10^{6}	2.00	1.511×10^{-7}
Fully Connected	2.57×10^{6}	7.22×10^{6}	2.00	0
Tree	3.60×10^{6}	2.29×10^{6}	2.00	4.69×10^{-4}

Table 5.3: Resulting coefficients and RMSE from fitting the model given by (5.3) to the total latency simulation results using the models derived in Section 4.3 for the various pipelines

while for the tree pipeline $a > b$. Consequently, the former are (slightly) more sensitive to changes in BW whereas the latter are more sensitive to IPS.

Observe that the intercept c is exactly equal to the window length. This is expected as this part of the latency cannot be reduced by faster processing and communication.

5.5.3 Conclusions

When minimizing the energy consumption, the biggest effect of a change in the efficiency of the components is to be gained from the communication stack. Consequently, if the efficiencies of both processor and communication stack scale equally with e.g. the size or cost, resources should go towards improving the communication stack. However, it is unlikely that both will scale equally and as such, the hardware scaling should be combined with the scaling behaviour from this section to optimize the implementation.

The impact of the bandwidth and processor speed on the latency is approximately equal. Hence the hardware scaling will determine how to best optimize the hardware for hitting the latency target.

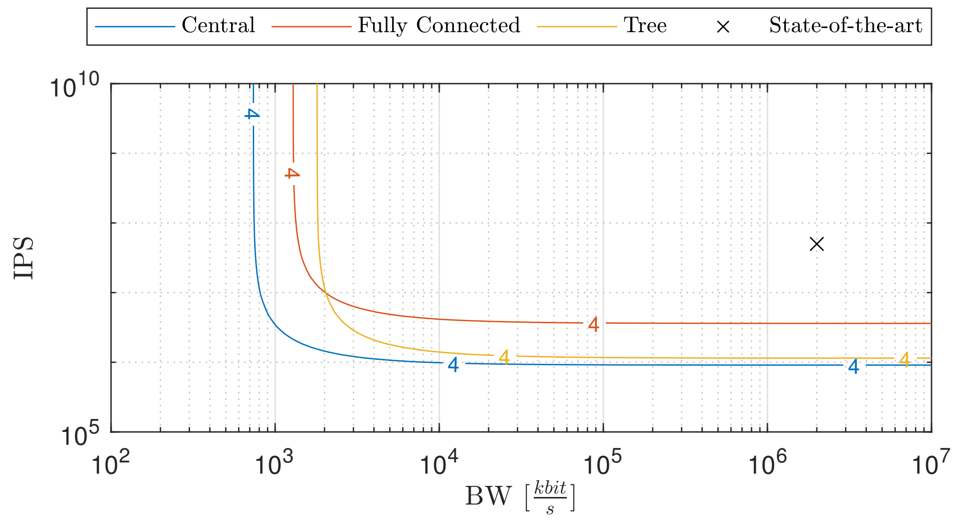


FIGURE 5.8: Contours of the latency i.f.o. the communication bandwidth and processor speed for both the centralized and distributed pipelines. Also indicated is the current state-of-the-art bandwidth and processor speed, as used with the performance models, given in Table 5.1.

5.6 Discussion

At a sufficiently large scale both in terms of number of nodes as well as channels per node, the distributed approaches improve the performance beyond that of the naive centralized approach and with better scaling behaviour in terms of the number of channels per node as well as network size. Indeed, when repeating the experiments on topologies similar in size to those in [14], the distributed pipelines outperform the centralized one.

However, the results of the first experiment show that for the application at hand, the optimal size of the network as well as the average number of channels per node is smaller than considered in [14]. Increasing these sizes to be similar was shown to not yield increased accuracy while it does result in a higher energy consumption as well as a higher latency and in making the system more complicated to use. At this smaller scale, the added overhead of the distributed pipelines are not outweighed by decreases in energy consumption and latency and as such, the distributed pipelines are surpassed in performance by the centralized approach.

Furthermore, in [14] it was also stated that in order for a node to have a battery life of 30 days, the power consumption should be $<140\,\mu W$. The base topology has an energy consumption of approximately 40 mJ per 2 second window, resulting in a continuous power consumption of 20 mW. Hence aside from the distributed approaches not being able to lower the energy consumption, the centralized pipeline's energy consumption still results in unsatisfactory battery life figures.

In applications where the network is larger, the experiments corroborate the performance results of the distributed approaches found in literature and these approaches can therefore be useful in such larger networks. For example in epileptic surgery, source localization can be used to locate the part of the brain responsible for pharmacoresistant seizures [41]. Here it makes sense to cover the entire scalp with nodes, resulting in a larger network. Furthermore, whereas the interelectrode distance in a BCI must be $>3\,\text{cm}$, other applications could benefit from more dense recordings, resulting in either further miniaturized nodes or in a higher channel count per electrode.

The last two experiments provides tools to design and optimize a hardware implementation for the envisioned nodes. The simplicity of the linear models allows to quickly assess the impact of a design change.

Chapter 6

Conclusion

Long-term neural monitoring has the potential to enable a range of diagnostic, instrumental and rehabilitative applications. However, as the traditional EEG recording setup is not suited to continuous, wearable and long term use, a novel approach was recently proposed that uses a WESN instead, specifically designed with these use cases in mind. To overcome the stringent energy requirements of this architecture, distributed signal processing algorithms were also proposed that replace typical algorithms that would require too much communication which would yield an unacceptably short battery life. So far however, this approach has not yet been applied as part of complete system for a specific application and as such, the performance and feasibility of such a system is still unknown.
The goal of this thesis was therefore to evaluate the performance and feasibility of such a complete system for use as a BCI based on the MI-paradigm, based on three goals: maximum accuracy, minimal energy consumption and minimal latency.

To this end, the DACGEE algorithm was integrated into a MI classification pipeline as an adaptation of the popular CSP feature extraction method. In order to also be used in a tree topology, the algorithm was also extended based on previous work by its authors. Furthermore, the algorithms were extended to efficiently perform multiple iterations for a single window of data, to vary the convergence speed. Lastly, as CSP is known to be prone to overfitting, the algorithms were re-derived to include Thikhonov regularization.
To do the evaluation, two techniques were employed: modelling and emulation. For the latency and energy consumption, models were created based on the classification pipelines' specifications. To estimate the decoding accuracy, data as it would be recorded by a WESN was emulated based on previously recorded HD-EEG data.

The size of the WESN was found to be a balance between low energy consumption and low latency with poor decoding accuracy for small networks, and higher accuracy but with a higher energy consumption and latency for larger networks. Going beyond 8 nodes yielded no additional improvements in decoding accuracy.

Due to the small size of the topologies applicable for the considered use case, the naive, centralized pipeline outperformed the distributed pipelines for all three of the considered metrics. Hence the centralized pipeline is preferred.
Important for designing a hardware implementation of a node, it was found that there exists a maximum possible number of channels per node that can simultaneously be used by the system. Also, the impact of the technological parameters was summarized into a linear model for the influence on the latency and one for energy consumption.

There are 3 main contributions of this thesis: extensions of the distributed algorithms, integration of the distributed algorithms in an actual application, and the evaluation. Belonging to the former, is the extension of efficient re-use to tree topologies, as it was only presented for fully connected topologies in [47]. A more substantial contribution are the re-derivations of the algorithms to include Thikhonov regularization, based on the compression of the filter matrix norm in [46]. This is potentially also useful for including regularization in distributed algorithms based on the same principle (see [21]).
The integration of the distributed algorithms into an actual application posed integration challenges such as how to perform the initialization and how to always have data from both classes available. Other applications based on similar distributed architectures are likely to face the same challenges.

6.1 Future work

Prime candidates for future consideration are the limitations of the analysis presented in this thesis:

- As this thesis does not explicitly focus on achieving maximum accuracy, the decoding accuracy can potentially be improved by looking at the following aspects:
  - Optimal design and placement of the nodes
  - Subject-specific tuning of the parameters
  - Use TRO instead of a GEVD
- As a simplification, the actual label was used to decide which covariance matrix to update. This is however not realistic and hence the decoded label should be used instead, potentially with some mechanism to repeat cue such that the accuracy is increased (in exchange for a lower information transfer rate).
- The distributed algorithms are synchronous, requiring some sort of synchronization between the nodes. Asynchronous algorithms would make this unnecessary, with the potential to have different speeds in the same network.
- Use a hybrid BCI that exploits multiple paradigms simultaneously, e.g. MI + BP

Bringing long-term neural monitoring systems into reality, poses new challenges, different from those found in the lab environment.

When the run time of the systems are extended far beyond the length of a typical lab experiment, the non-perfect stationarity of the EEG statistics mandates the use of an adaptive system. However, thus far there exists no way of measuring the rate at which the statistics change, nor a consensus as to what method of adaptation is best suited and how this interacts with feature extraction algorithms. A study into this topic will not only be of use to WESNs, but to all paradigms that aim to break BCI's out of the lab.

Currently the best performance is obtained from systems that are specifically tuned to each user. However, tuning requires expert assistance. If a system can be made that is independent of the characteristics of a specific user, or if the fitting to a specific user is either automated (adaptive) or user-performable, it will be easier, more cost effective and less time consuming to implement in clinical practice. Related is the robust placement of the sensor nodes or more generally electrodes. A method of selecting the correct placement, either physical or virtual placement trough electrode selection, could further improve the ease of setting up the system.

The most obvious continuation is the development of a hardware implementation of the nodes as envisioned in this thesis. Also, the design of a suitable wireless stack that is both lightweight (as it needs to fit on low power nodes) but at the same time flexible, robust and secure is crucial to creating systems that can be used independently by lay users for longer periods of time. Together with a suitable wireless stack, the system running on top of the hardware should be designed to efficiently work together with the wireless stack for optimal performance. For example, algorithms can be made aware of node outages or additions and use this information to more efficiently use the underlying network. Furthermore, a protocol is needed that controls what happens when a misclassification is detected.

Appendix A

Derivation of performance models

Sections 4.1 and 4.2 discussed methods to estimate respectively the computational complexity and the communication bandwidth and applied them to all pipelines given in chapter 3. However, the details of applying these methods to the pipelines were omitted from the main text for the sake of conciseness and easy exposition in the main text and are instead presented in this appendix

In the first section, the details of the derivation of the computational complexity models are presented. The second section then covers those of the communication bandwidth models.

A.1 Computational complexity models

In this section, the computational complexity of all the pipelines that are proposed in this work is determined, per window and per node, using the method discussed in section 4.1:

Step 1: Determine the complexity of each step and accumulate
Step 2: Simplify the sum using typical network parameters as given in Table 4.1
Step 3: For distributed pipelines: Determine the relative frequency of each case
Step 4: For distributed pipeline: Use the relative frequency to compute the mean complexity
Step 5: Simply the result further by using upper bounds

A.1.1 Centralized algorithm on a WESN

Table A.1 then gives the computational complexity for each of the steps of Algorithm 3.1. If implemented on a WESN, only the central node k_{FCe} needs to do processing. Hence the complexity model only holds for that central node.

Step	Complexity (if $k = k_{FCe}$)
Filtering	QMN
Calculate features	QN
Update covariance matrix	$(L-1)M^2N$
Calculate optimal filter (GEVD)	M^3
Resolve sign ambiguity	MQ
Total	$M^3 + NM(Q + (L-1)M) + (N+M)Q$ $\approx M^3 + (L-1)NM^2$ $< M^3 + LNM^2$

Table A.1: Computational complexity of the steps of 1 iteration of the centralized distributed pipeline (Alg. 3.1)

A.1.2 Distributed pipelines without efficient re-use

For the proposed pipelines given in Alg. 3.8 and 3.9 the complexity is similarly derived but now the distinction is made between the updating node q (i.e. the node that solves the compressed optimal filtering problem in that iteration) and the rest of the nodes. The computational complexity models for the steps of these pipelines are given in Table A.2 and A.3 respectively.

As each iteration the node that solves the compressed problem rotates between the K nodes, the relative frequency p(k) for a node to be in one of the two cases can easily be seen to be:

$$p(k) = \begin{cases} \frac{1}{K} & \text{if } k = q \\ \frac{K-1}{K} & \text{if } k \neq q \end{cases} . \tag{A.1}$$

Hence on average, the computational complexity for a single node and window is

Step	Complexity
Filtering	QM_qN
Calculate features	NQK
Update covariance matrix	$(L-1)\widetilde{M_q}^2N$
Calculate optimal filter (GEVD)	$\widetilde{M_q}^3$
Resolve sign ambiguity	M_qQ
Total	$\widetilde{M_q}^2(\widetilde{M_q}+(L-1)N)$ $+ M_qQ(1+N)+NQK$ $\approx \widetilde{M_q}^3+(L-1)\widetilde{M_q}^2N$

(A) In node q which updates the filters

Step	Complexity
Calculate L_k^i	M_kQ^2
Filtering	QM_kN
Calculate features	NQK
Update covariance matrix	$(L-1)\widetilde{M_k}^2N$
Update filter matrix	M_kQ^2
Total	$QN((L-1)M_k+K)+Q^2M_k+\widetilde{M_k}^2N$ $\approx (L-1)\widetilde{M_k}^2N$

(B) In non-updating node k

Table A.2: Computational complexity of the steps of 1 iteration i of the distributed pipeline in fully connected topologies without re-use (Alg. 3.8)

given by the following equations for Algs. 3.8 and 3.9 respectively:

$$\begin{aligned}
E\left[N_{comp,k}\right] &\approx \frac{1}{K}\left(\widetilde{M_k}^3+(L-1)\widetilde{M_k}^2N\right)+\frac{K-1}{K}\left(\widetilde{M_k}^2N\right) && \text{(A.2)}\\
&\approx (L-1)\widetilde{M_k}^2N && \text{(A.3)}\\
&< L\widetilde{M_k}^2N\,, && \text{(A.4)}\\
E\left[N_{comp,k}\right] &\approx \frac{1}{K}\left(\widetilde{M_k}^3+(L-1)\widetilde{M_k}^2N+QN(M_k+|\mathcal{N}_k|^2+|\mathcal{N}_k|)\right)\\
&\quad +\frac{K-1}{K}\left((L-1)\widetilde{M_k}^2N+QN(M_k+|\mathcal{N}_k|)\right) && \text{(A.5)}\\
&\approx (L-1)\widetilde{M_k}^2N+QN(M_k+|\mathcal{N}_k|) && \text{(A.6)}\\
&< L\widetilde{M_k}^2N+QN(M_k+|\mathcal{N}_k|)\,. && \text{(A.7)}
\end{aligned}$$

A.1.3 Distributed pipelines with efficient re-use

Finally, for Pipelines 3.10 and 3.11, another distinction is made, apart from the one from the previous pipeline. Indeed, whereas in the previously discussed pipelines, one iteration i corresponded to one unique window of data, the last two pipelines re-use one window for R iterations $i, \dots, i + R - 1$. To re-use the window in an efficient manner, the steps taken are different in the cases in which $r = 1$, $r = R$ and $1 < r < R$ (see Section 3.5).
Table A.4 and A.5 then give the computational complexity per iteration i for Pipelines 3.10 and 3.11 respectively, where Subtable a and b give the complexity for an updating node and non-updating node in that iteration respectively and it is also indicated which steps need not be done for certain values of r, thereby accounting for all possible cases.

Similarly to the previous two discussed pipelines, the mean complexity can be determined by weighing each case with its relative frequency.
In general, the tuple (r, q) visits every element in the product set $\{1, \dots, R\} \times \mathcal{K}$. If it does, the relative frequency of all of the elements in that set is the same and equal to $\frac{1}{RK}$. As the function $i \to (r, q)$ is indeed a function (in the sense that it maps i to only one specific tuple (r, q)), the relative frequency of a subset of $\{1, \dots, R\} \times \mathcal{K}$ is equal to the cardinality of that subset multiplied with $\frac{1}{RK}$.
For the different cases possible in an iteration of Algorithms 3.10 and 3.11, the relative frequency $p(k, r)$ can be summarized as follows:

$$\begin{array}{c|ccc}
 & r = 1 & 1 < r < R & r = R \\
\hline
k = q & \frac{1}{RK} & \frac{R-2}{RK} & \frac{1}{RK} \\
k \neq q & \frac{K-1}{RK} & \frac{(R-2)(K-1)}{RK} & \frac{K-1}{RK}
\end{array} \tag{A.8}$$

The mean complexity is then given by

$$\begin{aligned}
E\left[N_{comp,k}\right] \approx\ & \frac{1}{RK}\left(\widetilde{M_k}^3 + (L-1)\widetilde{M_k}^2 N + (K-1)Q^2 N\right) \\
& + \frac{R-2}{RK}\left(\widetilde{M_k}^3 + (L-1)\widetilde{M_k}^2 N + (K-1)Q^2 N\right) \\
& + \frac{1}{RK}\left(\widetilde{M_k}^3 + (L-1)\widetilde{M_k}^2 N\right) \\
& + \frac{K-1}{RK}\left((L-1)\widetilde{M_k}^2 N + (K-1)Q^2 N\right) \qquad \text{(A.9)} \\
& + \frac{(R-2)(K-1)}{RK}\left((L-1)\widetilde{M_k}^2 N + (K-1)Q^2 N\right) \\
& + \frac{K-1}{RK}\left((L-1)\widetilde{M_k}^2 N\right) \\
=\ & (L-1)\widetilde{M_k}^2 N + \frac{1}{K}\widetilde{M_k}^3 + \frac{R-1}{R}(K-1)Q^2 N \qquad \text{(A.10)} \\
<\ & L\widetilde{M_k}^2 N + \frac{1}{K}\widetilde{M_k}^3 + RKQ^2 N \qquad \text{(A.11)} \\
\approx\ & L\widetilde{M_k}^2 N + RKQ^2 N \qquad \text{(A.12)}
\end{aligned}$$

for Algorithm 3.10, and for Algorithm 3.11 by

$$\begin{aligned}
E\left[N_{comp,k}\right] \approx\ & \frac{1}{RK}\left(\widetilde{M_k}^3 + (L-1)\widetilde{M_k}^2 N + QN(M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k|) + |\mathcal{N}_k|Q^2 N\right) \\
& + \frac{R-2}{RK}\left(\widetilde{M_k}^3 + (L-1)\widetilde{M_k}^2 N + QN(M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k|) + |\mathcal{N}_k|Q^2 N\right) \\
& + \frac{1}{RK}\left(\widetilde{M_k}^3 + (L-1)\widetilde{M_k}^2 N + QNM_k\right) \qquad \text{(A.13)} \\
& + \frac{K-1}{RK}\left((L-1)\widetilde{M_k}^2 N + QN(M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k|) + n_k Q^2 N\right) \\
& + \frac{(R-2)(K-1)}{RK}\left((L-1)\widetilde{M_k}^2 N + QN(M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k|) + |\mathcal{N}_k|Q^2 N\right) \\
& + \frac{K-1}{RK}\left((L-1)\widetilde{M_k}^2 N + QNM_k\right) \\
=\ & (L-1)\widetilde{M_k}^2 N + QNM_k + \frac{1}{K}\widetilde{M_k}^3 \\
& + \frac{R-1}{R}\left(QN(M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k| + |\mathcal{N}_k|Q)\right) \qquad \text{(A.14)} \\
<\ & L\widetilde{M_k}^2 N + QNM_k + \frac{1}{K}\widetilde{M_k}^3 + QN(M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k| + |\mathcal{N}_k|Q) \\
& \qquad \text{(A.15)} \\
\approx\ & L\widetilde{M_k}^2 N + QN(2M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k| + |\mathcal{N}_k|Q) \qquad \text{(A.16)}
\end{aligned}$$

From these equations, the mean complexity per window rather than per iteration immediately follows. Since there are R iterations per window, equations (A.12) and (A.16) need to be multiplied with R:

$$E\left[N_{comp,k}\right] \lessapprox LR\widetilde{M_k}^2 N + KQ^2N\,, \tag{A.17}$$

$$E\left[N_{comp,k}\right] \lessapprox LR\widetilde{M_k}^2 N + QN(2M_k + |\mathcal{N}_k|^2 + |\mathcal{N}_k| + |\mathcal{N}_k|Q)\,. \tag{A.18}$$

Step	Complexity
Filtering	QM_qN
Calculate $\bar{\mathbf{x}}^i_{q\to n_l}$, $\forall n_l \in \mathcal{N}_q$	$\lvert\mathcal{N}_q\rvert QN(\lvert\mathcal{N}_q\rvert - 2)$
Calculate features	$QN\lvert\mathcal{N}_q\rvert$
Update covariance matrix	$(L-1)\widetilde{M_q}^2N$
Calculate optimal filter (GEVD)	$\widetilde{M_q}^3$
Resolve sign ambiguity	M_qQ
Total	$\widetilde{M_q}^2(\widetilde{M_q} + (L-1)N) + M_qQ(1+N) + \lvert\mathcal{N}_q\rvert^2QN + QNn_q \approx \widetilde{M_q}^3 + (L-1)\widetilde{M_q}^2N + QN(M_q + \lvert\mathcal{N}_q\rvert^2 + \lvert\mathcal{N}_q\rvert)$

(A) In node q which updates the filters

Step	Complexity
Calculate $\mathbf{L}^i_k$	M_kQ^2
Filtering	QM_kN
Calculate $\mathbf{L}^i_{k\to l_{uplink}}$	$\lvert\mathcal{N}_k\rvert Q^2(\lvert\mathcal{N}_k\rvert - 2)$
Calculate $\bar{\mathbf{x}}^i_{k\to n_l}$, $\forall n_l \in \mathcal{N}_k$	$\lvert\mathcal{N}_k\rvert QN(\lvert\mathcal{N}_k\rvert - 2)$
Calculate features	$QN\lvert\mathcal{N}_k\rvert$
Update covariance matrix	$(L-1)\widetilde{M_k}^2N$
Update filter matrix	M_kQ^2
Total	$QN(M_k + \lvert\mathcal{N}_k\rvert) + Q^2M_k + \widetilde{M_k}^2N + \lvert\mathcal{N}_k\rvert^2Q^2 \approx \widetilde{M_k}^2N + QN(M_k + \lvert\mathcal{N}_k\rvert)$

(B) In non-updating node k

Table A.3: Computational complexity of the steps of 1 iteration i of the distributed pipeline in tree topologies without re-use (Alg. 3.9)

Step	Complexity
Filtering (if $r = 1$)	QM_qN
Calculate features	NQK
Update covariance matrix	$(L-1)\widetilde{M_q}^2N$
Calculate optimal filter (GEVD)	$\widetilde{M_q}^3$
Resolve sign ambiguity	M_qQ
Filtering (if $r \neq R$)	QM_qN
Update received compressed observations (if $r \neq R$)	$(K-1)Q^2N$

(A) In node q which updates the filters

Step	Complexity
Calculate $\mathbf{L}_k^i$	M_kQ^2
Filtering (if $r = 1$)	QM_kN
Calculate features	NQK
Update covariance matrix	$(L-1)\widetilde{M_k}^2N$
Update filter matrix	M_kQ^2
Update received compressed observations (if $r \neq R$)	$(K-1)Q^2N$

(B) In non-updating node k

TABLE A.4: Computational complexity of 1 iteration i of the distributed pipeline in fully connected networks with data-reuse (Alg. 3.10)

Step	Complexity
Filtering (if $r = 1$)	QM_qN
Calculate $\bar{\mathbf{x}}^i_{q\to n_l}$, $\forall n_l \in \mathcal{N}_q$ (if $r = 1$)	$\lvert\mathcal{N}_q\rvert QN(\lvert\mathcal{N}_q\rvert - 1)$
Calculate features	$NQ\lvert\mathcal{N}_q\rvert$
Update covariance matrix	$(L-1)\widetilde{M_q}^2N$
Calculate optimal filter (GEVD)	$\widetilde{M_q}^3$
Resolve sign ambiguity	M_qQ
Filtering (if $r \neq R$)	QM_qN
Update received compressed observations (if $r \neq R$)	$\lvert\mathcal{N}_q\rvert Q^2N$
Calculate $\bar{\mathbf{x}}^{i+1}_{q\to n_l}$, $\forall n_l \in \mathcal{N}_q$ (if $r \neq R$)	$\lvert\mathcal{N}_q\rvert^2QN$

(A) In node q which updates the filters

Step	Complexity
Filtering (if $r = 1$)	QM_kN
Calculate $\bar{\mathbf{x}}^i_{k\to n_l}$, $\forall n_l \in \mathcal{N}_k$ (if $r = 1$)	$(\lvert\mathcal{N}_k\rvert QN(\lvert\mathcal{N}_k\rvert - 1)$
Calculate $\mathbf{L}^i_k$	M_kQ^2
Calculate $\mathbf{L}^i_{k\to l_{uplink}}$	$Q^2(\lvert\mathcal{N}_k\rvert - 1)$
Calculate features	$NQ(\lvert\mathcal{N}_k\rvert$
Update covariance matrix	$(L-1)\widetilde{M_k}^2N$
Update filter matrix	M_kQ^2
Update own compressed observations (if $r \neq R$)	Q^2N
Update received compressed observations (if $r \neq R$)	$(\lvert\mathcal{N}_k\rvert - 1)Q^2N$
Calculate $\bar{\mathbf{x}}^{i+1}_{k\to n_l}$, $\forall n_l \in \mathcal{N}_k$ (if $r \neq R$)	$(\lvert\mathcal{N}_k\rvert - 1)^2QN$

(B) In non-updating node k

TABLE A.5: Computational complexity of 1 iteration i of the distributed pipeline in tree networks with data-reuse (Alg. 3.11)

A.2 Communication bandwidth models

In this section, the required communication bandwidth of all the pipelines that are proposed in this work is determined, per window and per node, according to the method discussed in Section 4.2:

Step 1:	Determine the number of words in each of the variables that need to be transmitted in one window, and sum
Step 2:	Simplify the sum using typical network parameters as given in Table 4.1
Step 3:	For distributed algorithms: Determine the relative frequency of each case
Step 4:	For distributed algorithms: Use the relative frequency to compute the mean
Step 5:	Simply the result further by using upper bounds

A.2.1 Centralized algorithm on a WESN

Table A.6 then gives the number of words in each of the variables that need to be transmitted (Subtable a) and received (Subtable b) in one iteration of the centralized pipeline (Alg. 3.1).

Variable	$k \neq c$	$k = c$
$\mathbf{x}_k^i$	NM_k	-

(A) Transmission

Variable	$k \neq c$	$k = c$
$\left\{\mathbf{x}_j^i\right\}_{k \in \mathcal{K} \setminus c}$	-	$(M - M_c)N$

(B) Reception

Table A.6: Communicational complexity of 1 iteration of the centralized distributed pipeline (Alg. 3.1)

Note that similarly to the computational complexity model, the required bandwidth is very different for the central node ($k = c$) and the peripheral nodes ($k \neq c$).

A.2.2 Distributed algorithms

For Pipelines 3.8 and 3.9 the number of words for each of the transmitted and received variables is similarly derived. The results are shown in Tables A.7 and A.8 where the distinction was made between an updating node ($k = q$) and non-updating node ($k \neq q$), identical to how it was done in the previous section for the complexity model.

The total amount of words per window, i.e. the sum of the columns in Table A.7,

Variable	$k \neq q$	$k = q$
$\bar{\mathbf{x}}_k^i$	NQ	NQ
$\mathbf{L}_k^i$	$\frac{Q(Q+1)}{2}$	-
$\mathbf{G}_{-q}$	-	$(K-1)Q^2$

(A) Transmission

Variable	$k \neq q$	$k = q$
$\left\{\bar{\mathbf{x}}_j^i\right\}_{j \in \mathcal{K} \setminus \{k\}}$	$(K-1)NQ$	$(K-1)NQ$
$\left\{\mathbf{L}_j^i\right\}_{j \in \mathcal{K} \setminus \{k\}}$	-	$(K-1)\frac{Q(Q+1)}{2}$
$\mathbf{G}_k$	Q^2	-

(B) Reception

Table A.7: Communicational complexity of 1 iteration i of the distributed pipeline in fully connected topologies without re-use (Alg. 3.8)

Variable	$k \neq q$	$k = q$
$\left\{\bar{\mathbf{x}}_{k \to n_l}^i\right\}_{n_l \in \mathcal{N}_k}$	$\lvert\mathcal{N}_k\rvert NQ$	$\lvert\mathcal{N}_k\rvert NQ$
$\mathbf{L}_{k \to l_{uplink}}^i$	$\frac{Q(Q+1)}{2}$	-
$\left\{\mathbf{G}_j\right\}_{j \in \mathcal{N}_k}$	-	$\lvert\mathcal{N}_k\rvert Q^2$

(A) Transmission

Variable	$k \neq q$	$k = q$
$\left\{\bar{\mathbf{x}}_{n_l \to k}^i\right\}_{n_l \in \mathcal{N}_k}$	$\lvert\mathcal{N}_k\rvert NQ$	$\lvert\mathcal{N}_k\rvert NQ$
$\left\{\mathbf{L}_{n_l \to k}^i\right\}_{n_l \in \mathcal{K} \setminus \{k\}}$	$(\lvert\mathcal{N}_k\rvert - 1)\frac{Q(Q+1)}{2}$	$\lvert\mathcal{N}_k\rvert \frac{Q(Q+1)}{2}$
$\mathbf{G}_k$	Q^2	-

(B) Reception

Table A.8: Communicational complexity of 1 iteration i of the distributed pipeline in tree topologies without re-use (Alg. 3.9)

can be simplified as follows for the fully connected case (Alg. 3.8):

$$N_{tx,k} = \begin{cases} NQ + (K-1)Q^2 < NQ + KQ^2 & \text{if } k = q \\ NQ + \frac{Q(Q+1)}{2} < NQ + Q^2 \approx NQ & \text{if } k \neq q \end{cases}, \tag{A.19}$$

$$N_{rx,k} = \begin{cases} (K-1)NQ + Q^2 < KNQ + Q^2 \approx KNQ & \text{if } k \neq q \\ (K-1)\left(NQ + \frac{Q(Q+1)}{2}\right) < K\left(NQ + Q^2\right) \approx KNQ & \text{if } k = q \end{cases}, \tag{A.20}$$

and similarly for the tree topology case (Table A.8) of Alg. 3.9:

$$N_{tx,k} = \begin{cases} |\mathcal{N}_k|NQ + \frac{Q(Q+1)}{2} < |\mathcal{N}_k|NQ + Q^2 \approx |\mathcal{N}_k|NQ & \text{if } k \neq q \\ |\mathcal{N}_k|NQ + |\mathcal{N}_k|Q^2 \approx |\mathcal{N}_k|NQ & \text{if } k = q \end{cases}, \tag{A.21}$$

$$N_{rx,k} = \begin{cases} |\mathcal{N}_k|NQ + (|\mathcal{N}_k| - 1)\frac{Q(Q+1)}{2} + Q^2 & \text{if } k \neq q \\ |\mathcal{N}_k|NQ + |\mathcal{N}_k|\frac{Q(Q+1)}{2} & \text{if } k = q \end{cases} \tag{A.22}$$

$$< |\mathcal{N}_k|NQ + |\mathcal{N}_k|Q^2 \approx |\mathcal{N}_k|NQ\,. \tag{A.23}$$

The relative frequency of the two cases a node can be in, was already derived in the previous section resulting in (A.1). The mean amount of words a node needs to transmit and receive can then be calculated, weighed by that relative frequency. For the fully connected case this gives:

$$E\left[N_{tx,k}\right] \lessapprox \frac{K-1}{K}NQ + \frac{1}{K}\left(NQ + KQ^2\right) = NQ + Q^2 \approx NQ\,, \tag{A.24}$$

$$E\left[N_{rx,k}\right] \lessapprox \frac{K-1}{K}KNQ + \frac{1}{K}KNQ = KNQ\,, \tag{A.25}$$

and for the tree topology case:

$$E\left[N_{tx,k}\right] \lessapprox \frac{K-1}{K}|\mathcal{N}_k|NQ + \frac{1}{K}|\mathcal{N}_k|NQ = |\mathcal{N}_k|NQ\,, \tag{A.26}$$

$$E\left[N_{rx,k}\right] \lessapprox \frac{K-1}{K}|\mathcal{N}_k|NQ + \frac{1}{K}|\mathcal{N}_k|NQ = |\mathcal{N}_k|NQ\,. \tag{A.27}$$

A.2.3 Distributed algorithms with efficient re-use

Finally for Pipelines 3.10 and 3.11 the same method is applied. In these pipelines, a distinction is not only made between updating and non-updating nodes, but also w.r.t. to how many times a window of data has already been used. This is identical to the approach taken with the complexity models for these pipelines (cf. supra). The results are given in Tables A.9 and A.10 for respectively the fully connected and tree topologies.

The total can then be calculated as the sum, and be further simplified as follows for

Variable	$k \neq q$	$k = q$
$\bar{\mathbf{x}}_k^i$ (if $r = 1$)	NQ	NQ
$\mathbf{L}_k^i$	$\frac{Q(Q+1)}{2}$	-
$\mathbf{G}_{-q}$	-	$(K-1)Q^2$
$\bar{\mathbf{x}}_k^{i+1}$ (if $r \neq R$)	-	NQ

(A) Transmission

Variable	$k \neq q$	$k = q$
$\left\{\bar{\mathbf{x}}_j^i\right\}_{j\in\mathcal{K}\backslash k}$ (if $r = 1$)	$(K-1)NQ$	$(K-1)NQ$
$\left\{\mathbf{L}_j^i\right\}_{j\in\mathcal{K}\backslash k}$	-	$(K-1)\frac{Q(Q+1)}{2}$
$\mathbf{G}_{-q}$	$(K-1)Q^2$	-
$\bar{\mathbf{x}}_q^{i+1}$ (if $r \neq R$)	NQ	-

(B) Reception

Table A.9: Communicational complexity of 1 iteration i of the distributed pipeline in fully connected networks with data-reuse (Alg. 3.10)

fully connected topologies:

$$N_{tx,k\neq q} = \begin{cases} NQ + \frac{Q(Q+1)}{2} \approx NQ & \text{if } r = 1 \\ \frac{Q(Q+1)}{2} < Q^2 & \text{otherwise} \end{cases}, \tag{A.28}$$

$$N_{tx,k=q} = \begin{cases} 2NQ + (K-1)Q^2 < 2NQ + KQ^2 & \text{if } r = 1 \\ (K-1)Q^2 + NQ < NQ + KQ^2 & \text{if } 1 < r < R \\ (K-1)Q^2 < KQ^2 & \text{if } r = R \end{cases}, \tag{A.29}$$

$$N_{rx,k\neq q} = \begin{cases} KNQ + (K-1)Q^2 < KNQ + KQ^2 & \text{if } r = 1 \\ (K-1)Q^2 + NQ < NQ + KQ^2 & \text{if } 1 < r < R \\ (K-1)Q^2 < KQ^2 & \text{if } r = R \end{cases}, \tag{A.30}$$

$$N_{rx,k=q} = \begin{cases} (K-1)\left(NQ + \frac{Q(Q+1)}{2}\right) < KNQ + KQ^2 & \text{if } r = 1 \\ (K-1)\frac{Q(Q+1)}{2} < KQ^2 & \text{otherwise} \end{cases}, \tag{A.31}$$

Variable	$k \neq q$	$k = q$
$\left\{\bar{\mathbf{x}}^i_{k\to n_l}\right\}_{n_l\in\mathcal{N}_k}$ (if $r=1$)	$\|\mathcal{N}_k\|NQ$	$\|\mathcal{N}_k\|NQ$
$\mathbf{L}^i_{k\to l_{uplink}}$	$\frac{Q(Q+1)}{2}$	-
$\{\mathbf{G}_j\}_{j\in\mathcal{N}_k}$	-	$\|\mathcal{N}_k\|Q^2$
$\left\{\bar{\mathbf{x}}^{i+1}_{k\to n_l}\right\}_{n_l\in\mathcal{N}_k}$ (if $r\neq R$)	-	$\|\mathcal{N}_k\|NQ$
$\left\{\bar{\mathbf{x}}^{i+1}_{k\to n_l}\right\}_{n_l\in\mathcal{N}_k\setminus l_{uplink}}$ (if $r\neq R$)	$(\|\mathcal{N}_k\|-1)NQ$	-

(A) Transmission

Variable	$k \neq q$	$k = q$
$\left\{\bar{\mathbf{x}}^i_{n_l\to k}\right\}_{n_l\in\mathcal{N}_k}$ (if $r=1$)	$\|\mathcal{N}_k\|NQ$	$\|\mathcal{N}_k\|NQ$
$\left\{\mathbf{L}^i_{n_l\to k}\right\}_{n_l\in\mathcal{N}_k\setminus l_{uplink}}$	$(\|\mathcal{N}_k\|-1)\frac{Q(Q+1)}{2}$	-
$\left\{\mathbf{L}^i_{n_l\to k}\right\}_{n_l\in\mathcal{N}_k}$	-	$\|\mathcal{N}_k\|\frac{Q(Q+1)}{2}$
$\mathbf{G}_k$	Q^2	-
$\bar{\mathbf{x}}^{i+1}_{l_{uplink}\to k}$ (if $r\neq R$)	NQ	-

(B) Reception

Table A.10: Communication complexity of 1 iteration i of the distributed pipeline in tree networks with data-reuse (Alg. 3.11)

and for tree topologies:

$$N_{tx,k\neq q} = \begin{cases} NQ + \dfrac{Q(Q+1)}{2} + (|\mathcal{N}_k|-1)NQ \\ \qquad < |\mathcal{N}_k|NQ + Q^2 \approx |\mathcal{N}_k|NQ & \text{if } r=1 \\ \frac{Q(Q+1)}{2} + (|\mathcal{N}_k|-1)NQ < |\mathcal{N}_k|NQ + Q^2 \approx |\mathcal{N}_k|NQ & \text{if } 1<r<R \\ \frac{Q(Q+1)}{2} < Q^2 & \text{if } r=R \end{cases}, \tag{A.32}$$

$$N_{tx,k=q} = \begin{cases} |\mathcal{N}_k|NQ + |\mathcal{N}_k|Q^2 + |\mathcal{N}_k|NQ \approx 2|\mathcal{N}_k|NQ & \text{if } r=1 \\ |\mathcal{N}_k|Q^2 + |\mathcal{N}_k|NQ \approx |\mathcal{N}_k|NQ & \text{otherwise} \end{cases}, \tag{A.33}$$

$$N_{rx,k\neq q} = \begin{cases} |\mathcal{N}_k|NQ + (|\mathcal{N}_k|-1)\dfrac{Q(Q+1)}{2} + Q^2 + NQ \\ \qquad < (|\mathcal{N}_k|+1)NQ + |\mathcal{N}_k|Q^2 \approx (|\mathcal{N}_k|+1)NQ & \text{if } r=1 \\ (|\mathcal{N}_k|-1)\frac{Q(Q+1)}{2} + Q^2 + NQ < |\mathcal{N}_k|Q^2 + NQ & \text{if } 1<r<R \\ (|\mathcal{N}_k|-1)\frac{Q(Q+1)}{2} + Q^2 < |\mathcal{N}_k|Q^2 & \text{if } r=R \end{cases}, \tag{A.34}$$

$$N_{rx,k=q} = \begin{cases} |\mathcal{N}_k|NQ + |\mathcal{N}_k|\frac{Q(Q+1)}{2} \approx |\mathcal{N}_k|NQ & \text{if } r=1 \\ |\mathcal{N}_k|\frac{Q(Q+1)}{2} < |\mathcal{N}_k|Q^2 & \text{otherwise} \end{cases}. \tag{A.35}$$

(A.36)

The relative frequency of each of these cases is given by eq. (A.8) from the previous section. Hence the mean number of words a node transmit and receives per iteration can straightforwardly be calculated as follows for the fully connected topology:

$$\begin{aligned}
E\left[N_{tx,k}\right] \lessapprox\ & \frac{1}{RK}\left(2NQ + KQ^2\right) + \frac{R-2}{RK}\left(NQ + KQ^2\right) \\
& + \frac{1}{RK}\cdot KQ^2 + \frac{K-1}{RK}\cdot NQ \qquad\qquad (A.37) \\
& + \frac{(K-1)(R-2)}{RK}\cdot Q^2 + \frac{K-1}{RK}\cdot Q^2 \\
=\ & \frac{R+K-1}{RK}NQ + \frac{(K-1)(R-1)+RK}{RK}\cdot Q^2 \qquad\qquad (A.38) \\
<\ & \frac{R+K}{RK}NQ + 2Q^2 \qquad\qquad (A.39) \\
=\ & \left(\frac{1}{K}+\frac{1}{R}\right)NQ + 2Q^2\,, \qquad\qquad (A.40) \\
E\left[N_{rx,k}\right] \lessapprox\ & \frac{1}{RK}\left(KNQ + KQ^2\right) + \frac{R-2}{RK}\cdot KQ^2 \\
& + \frac{1}{RK}\cdot KQ^2 + \frac{K-1}{RK}\left(KNQ + KQ^2\right) \qquad\qquad (A.41) \\
& + \frac{(K-1)(R-2)}{RK}\left(NQ + KQ^2\right) + \frac{K-1}{RK}\cdot KQ^2 \\
=\ & \left(\frac{K^2+(R-2)(K-1)}{RK}\right)NQ + Q^2 \qquad\qquad (A.42) \\
<\ & \left(\frac{K^2+RK}{RK}\right)NQ + Q^2 = \left(1+\frac{K}{R}\right)NQ + Q^2 \qquad\qquad (A.43) \\
\approx\ & \left(1+\frac{K}{R}\right)NQ\,, \qquad\qquad (A.44)
\end{aligned}$$

and for tree topologies:

$$\begin{aligned}
E\left[N_{tx,k}\right] &\lessapprox \frac{1}{RK}\cdot 2|\mathcal{N}_k|NQ + \frac{R-1}{RK}\cdot|\mathcal{N}_k|NQ \\
&\quad + \frac{K-1}{RK}\cdot|\mathcal{N}_k|NQ + \frac{(K-1)(R-2)}{RK}\cdot|\mathcal{N}_k|NQ + \frac{K-1}{RK}\cdot Q^2 && \text{(A.45)} \\
&= \frac{2+R(K-1)}{RK}\cdot|\mathcal{N}_k|NQ + \frac{K-1}{RK}\cdot Q^2 && \text{(A.46)} \\
&\approx \frac{K-1}{K}\cdot|\mathcal{N}_k|NQ + \frac{1}{R}\cdot Q^2 && \text{(A.47)} \\
&\approx |\mathcal{N}_k|NQ\,, && \text{(A.48)} \\
E\left[N_{rx,k}\right] &\lessapprox \frac{1}{RK}\cdot|\mathcal{N}_k|NQ + \frac{R-1}{RK}\cdot|\mathcal{N}_k|Q^2 + \frac{K-1}{RK}\cdot(|\mathcal{N}_k|+1)NQ \\
&\quad + \frac{(R-2)(K-1)}{RK}\cdot(|\mathcal{N}_k|Q^2 + NQ) + \frac{K-1}{RK}\cdot|\mathcal{N}_k|Q^2 && \text{(A.49)} \\
&= \frac{K(R+|\mathcal{N}_k|-1)-R+1}{RK}\cdot NQ + \frac{R(K-1)}{RK}\cdot Q^2 && \text{(A.50)} \\
&< \frac{(R+|\mathcal{N}_k|)}{R}\cdot NQ + Q^2 && \text{(A.51)} \\
&\approx \frac{(R+|\mathcal{N}_k|)}{R}\cdot NQ\,. && \text{(A.52)}
\end{aligned}$$

As there are R iterations per window, the means per window become:

$$\begin{aligned}
E\left[N_{tx,k}\right] &\lessapprox \left(\frac{R}{K}+1\right)NQ + 2RQ^2\,, && \text{(A.53)} \\
E\left[N_{rx,k}\right] &\lessapprox (R+K)NQ\,, && \text{(A.54)}
\end{aligned}$$

for fully connected topologies. Similarly for tree topologies:

$$\begin{aligned}
E\left[N_{tx,k}\right] &\lessapprox |\mathcal{N}_k|RNQ\,, && \text{(A.55)} \\
E\left[N_{rx,k}\right] &\lessapprox (R+|\mathcal{N}_k|)NQ\,. && \text{(A.56)}
\end{aligned}$$

Bibliography

[1] A. Berk, C. A. Kaiser, H. Lodish, A. Amon, H. Ploegh, A. Bretscher, M. Krieger, and K. C. Martin, “Ch. 22: Cells of the Nervous System,” in *Molecular Cell Biology*. Macmillan Learning, Apr. 2016, pp. 1025–1077.

[2] G. L. Holmes and R. Khazipov, “Basic Neurophysiology and the Cortical Basis of EEG,” in *The Clinical Neurophysiology Primer*, A. S. Blum and S. B. Rutkove, Eds. Totowa, NJ: Humana Press, 2007, pp. 19–33.

[3] BruceBlaus, “An illustration depicting a neuron.” Jun. 2017.

[4] T. Sheerman-Chase, “EEG Brain Scan,” Oct. 2012.

[5] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-r. Muller, “Optimizing Spatial filters for Robust EEG Single-Trial Analysis,” *IEEE Signal Processing Magazine*, vol. 25, no. 1, pp. 41–56, 2008.

[6] R. Tomioka, J. Hill, B. Blankertz, and K. Aihara, “Adapting Spatial Filtering Methods for Nonstationary BCIs,” in *2006 Workshop on Information-Based Induction Sciences (IBIS2006)*, Osaka, Japan, Nov. 2006, p. 6.

[7] S. Sun and C. Zhang, “Adaptive feature extraction for EEG signal classification,” *Medical & Biological Engineering & Computing*, vol. 44, no. 10, pp. 931–935, Oct. 2006.

[8] P. Shenoy, M. Krauledat, B. Blankertz, R. P. N. Rao, and K.-R. Müller, “Towards adaptive classification for BCI,” *Journal of Neural Engineering*, vol. 3, no. 1, pp. R13–R23, Mar. 2006.

[9] J. R. Wolpaw and C. B. Boulay, “Brain Signals for Brain–Computer Interfaces,” in *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*, ser. The Frontiers Collection, B. Graimann, G. Pfurtscheller, and B. Allison, Eds. Berlin, Heidelberg: Springer, 2010, pp. 29–46.

[10] B. Graimann, B. Allison, and G. Pfurtscheller, Eds., *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*, ser. The Frontiers Collection. Berlin Heidelberg: Springer-Verlag, 2010.

[11] G. Pfurtscheller and C. Neuper, “Dynamics of Sensorimotor Oscillations in a Motor Task,” in *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*, ser. The Frontiers Collection, B. Graimann, G. Pfurtscheller, and B. Allison, Eds. Berlin, Heidelberg: Springer, 2010, pp. 47–64.

[12] B. Blankertz, M. Tangermann, C. Vidaurre, T. Dickhaus, C. Sannelli, F. Popescu, S. Fazli, M. Danóczy, G. Curio, and K.-R. Müller, “Detecting Mental States by Machine Learning Techniques: The Berlin Brain–Computer Interface,” in *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*, ser. The Frontiers Collection, B. Graimann, G. Pfurtscheller, and B. Allison, Eds. Berlin, Heidelberg: Springer, 2010, pp. 113–135.

[13] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, “Brain–computer interfaces for communication and control,” *Clinical Neurophysiology*, p. 25, 2002.

[14] A. Bertrand, “Distributed Signal Processing for Wireless EEG Sensor Networks,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 6, pp. 923–935, Nov. 2015.

[15] A. Tiwari and A. Chaturvedi, “A Novel Channel Selection Method for BCI Classification Using Dynamic Channel Relevance,” *IEEE Access*, vol. 9, pp. 126 698–126 716, 2021.

[16] A. M. Narayanan and A. Bertrand, “Analysis of Miniaturization Effects and Channel Selection Strategies for EEG Sensor Networks With Application to Auditory Attention Detection,” *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 1, pp. 234–244, Jan. 2020.

[17] A. Mundanad Narayanan, R. Zink, and A. Bertrand, “EEG miniaturization limits for stimulus decoding with EEG sensor networks,” *Journal of Neural Engineering*, vol. 18, no. 5, p. 056042, Oct. 2021.

[18] J. Dan, S. Geirnaert, and A. Bertrand, “Grouped variable selection for generalized eigenvalue problems,” *Signal Processing*, vol. 195, p. 108476, Jun. 2022.

[19] T. Alotaiby, F. E. A. El-Samie, S. A. Alshebeili, and I. Ahmad, “A review of channel selection algorithms for EEG signal processing,” *EURASIP Journal on Advances in Signal Processing*, vol. 2015, no. 1, p. 66, Aug. 2015.

[20] A. Bertrand and M. Moonen, “Distributed adaptive generalized eigenvector estimation of a sensor signal covariance matrix pair in a fully connected sensor network,” *Signal Processing*, vol. 106, pp. 209–214, Jan. 2015.

[21] C. A. Musluoglu and A. Bertrand, “A Unified Framework for Distributed Signal and Feature Fusion Problems,” pp. 1–15.

[22] A. Bertrand, “Selected Topics in Biomedical Signal Processing: Part 1: Data-driven filter design for biomedical sensory arrays.”

[23] B. Blankertz, K.-R. Muller, D. Krusienski, G. Schalk, J. Wolpaw, A. Schlogl, G. Pfurtscheller, J. Millan, M. Schroder, and N. Birbaumer, “The BCI competition III: Validating alternative approaches to actual BCI problems,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 153–159, Jun. 2006.

[24] M. Tangermann, K.-R. Müller, A. Aertsen, N. Birbaumer, C. Braun, C. Brunner, R. Leeb, C. Mehring, K. Miller, G. Mueller-Putz, G. Nolte, G. Pfurtscheller, H. Preissl, G. Schalk, A. Schlögl, C. Vidaurre, S. Waldert, and B. Blankertz, “Review of the BCI Competition IV,” *Frontiers in Neuroscience*, vol. 6, p. 55, 2012.

[25] T. T. Ngo, M. Bellalij, and Y. Saad, “The Trace Ratio Optimization Problem,” *SIAM Review*, vol. 54, no. 3, pp. 545–569, Jan. 2012.

[26] S. Yan and X. Tang, “Trace Quotient Problems Revisited,” in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 3952, pp. 232–244.

[27] Y. Jia, F. Nie, and C. Zhang, “Trace Ratio Problem Revisited,” *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 729–735, Apr. 2009.

[28] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, “Trace Ratio vs. Ratio Trace for Dimensionality Reduction,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–8.

[29] S. Lemm, B. Blankertz, G. Curio, and K.-R. Muller, “Spatio-spectral filters for improving the classification of single trial EEG,” *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 9, pp. 1541–1548, Sep. 2005.

[30] A. Yuksel and T. Olmez, “Filter Bank Common Spatio-Spectral Patterns for Motor Imagery Classification,” in *Information Technology in Bio- and Medical Informatics*, M. E. Renda, M. Bursa, A. Holzinger, and S. Khuri, Eds. Cham: Springer International Publishing, 2016, vol. 9832, pp. 69–84.

[31] S. Geirnaert, T. Francart, and A. Bertrand, “Fast EEG-Based Decoding Of The Directional Focus Of Auditory Attention Using Common Spatial Patterns,” *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 5, pp. 1557–1568, May 2021.

[32] K. K. Ang, Z. Y. Chin, C. Wang, C. Guan, and H. Zhang, “Filter Bank Common Spatial Pattern Algorithm on BCI Competition IV Datasets 2a and 2b,” *Frontiers in Neuroscience*, vol. 6, p. 39, 2012.

[33] W. Samek, C. Vidaurre, K.-R. Müller, and M. Kawanabe, “Stationary common spatial patterns for brain–computer interfacing,” *Journal of Neural Engineering*, vol. 9, no. 2, p. 026013, Feb. 2012.

[34] A. Meinel, F. Lotte, and M. Tangermann, "Tikhonov Regularization Enhances Eeg-Based Spatial Filtering For Single-Trial Regression," in *Proceedings Of The 7Th Graz Brain-Computer Interface Conference 2017*. Graz University Of Technology, Austria: Verlag der Technischen Universität Graz, Sep. 2017.

[35] F. Lotte and C. Guan, "Regularizing Common Spatial Patterns to Improve BCI Designs: Unified Theory and New Algorithms," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 2, pp. 355–362, Feb. 2011.

[36] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, 2nd ed., ser. Springer Series in Statistics. New York: Springer-Verlag, 2009.

[37] K.-R. Muller, C. Anderson, and G. Birch, "Linear and nonlinear methods for brain-computer interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 165–169, Jun. 2003.

[38] "Linear discriminant analysis, explained · Xiaozhou's Notes."

[39] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. New York: Springer, 2006.

[40] J. Szurley, A. Bertrand, and M. Moonen, "Distributed adaptive node-specific signal estimation in heterogeneous and mixed-topology wireless sensor networks," *Signal Processing*, vol. 117, pp. 44–60, Dec. 2015.

[41] J. Wellmer, F. von der Groeben, U. Klarmann, C. Weber, C. E. Elger, H. Urbach, H. Clusmann, and M. von Lehe, "Risks and benefits of invasive epilepsy surgery workup with implanted subdural and depth electrodes," *Epilepsia*, vol. 53, no. 8, pp. 1322–1332, 2012.

[42] S. M. Lawrence, J. O. Larsen, K. W. Horch, R. Riso, and T. Sinkjær, "Long-term biocompatibility of implanted polymer-based intrafascicular electrodes," *Journal of Biomedical Materials Research*, vol. 63, no. 5, pp. 501–506, 2002.

[43] C. Marin and E. Fernandez, "Biocompatibility of intracortical microelectrodes: Current status and future prospects," *Frontiers in Neuroengineering*, vol. 3, 2010.

[44] B. Braem, B. Latre, I. Moerman, C. Blondia, E. Reusens, W. Joseph, L. Martens, and P. Demeester, "The Need for Cooperation and Relaying in Short-Range High Path Loss Sensor Networks," in *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, Oct. 2007, pp. 566–571.

[45] R. Cahn, *Wide Area Network Design: Concepts and Tools for Optimization*. Morgan Kaufmann, May 1998.

[46] C. A. Musluoglu and A. Bertrand, "Distributed Adaptive Trace Ratio Optimization in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3653–3670, 2021.

[47] C. A. Musluoglu, M. Moonen, and A. Bertrand, “Improved tracking for the distributed signal fusion optimization algorithm in a fully connected wireless sensor network (in press),” in *2022 30th European Signal Processing Conference (EUSIPCO)*, Belgrade, Serbia, Sep. 2022.

[48] A. Bertrand and M. Moonen, “Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA,” *Signal Processing*, vol. 104, pp. 120–135, Nov. 2014.

[49] D. P. Arbenz, “Lecture Notes on Solving Large Scale Eigenvalue Problems,” p. 265.

[50] T. Zasowski, G. Meyer, F. Althaus, and A. Wittneben, “Propagation effects in UWB body area networks,” in *2005 IEEE International Conference on Ultra-Wideband*, Sep. 2005, pp. 16–21.

[51] W. Joseph, B. Braem, E. Reusens, B. Latre, L. Martens, I. Moerman, and C. Blondia, “Design of Energy Efficient Topologies for Wireless On-Body Channel,” in *17th European Wireless 2011 - Sustainable Wireless Technologies*, Apr. 2011, pp. 1–7.

[52] D. Bresnahan and Y. Li, “Investigation of Creeping Wave Propagation Around the Human Head at ISM Frequencies,” *IEEE Antennas and Wireless Propagation Letters*, vol. 16, pp. 2767–2770, 2017.

[53] A. Mundanad Narayanan, “Miniaturization Effects and Node Placement for Neural Decoding in EEG Sensor Networks,” Ph.D. dissertation, Faculty of Engineering Science, KU Leuven, Leuven, Belgium, 2021.

[54] S. Galal and M. Horowitz, “Energy-Efficient Floating-Point Unit Design,” *IEEE Transactions on Computers*, vol. 60, no. 7, pp. 913–922, Jul. 2011.

[55] “Datasheet: nRF2401 - Single chip 2.4 GHz Transceiver,” Jun. 2004.