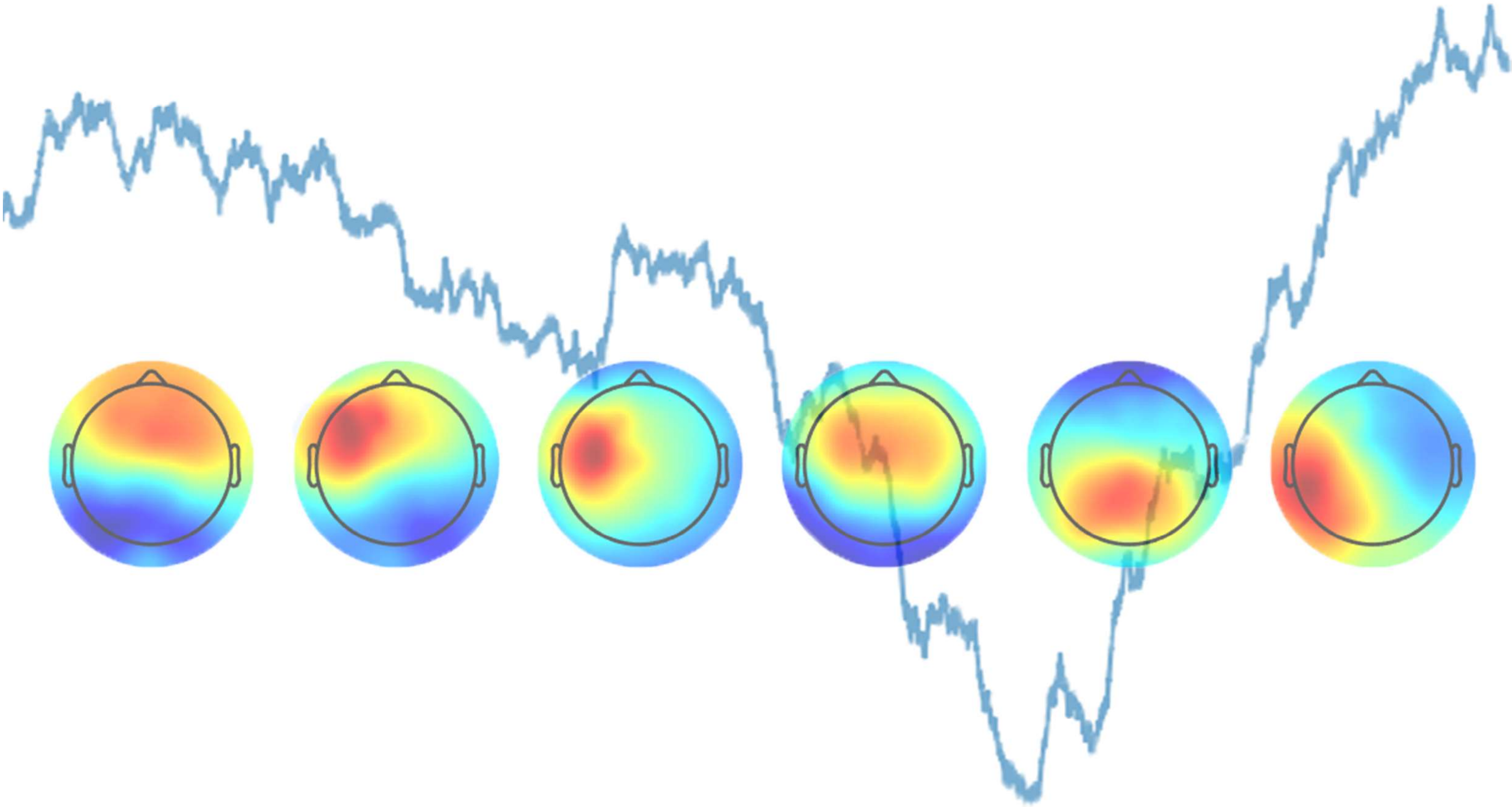
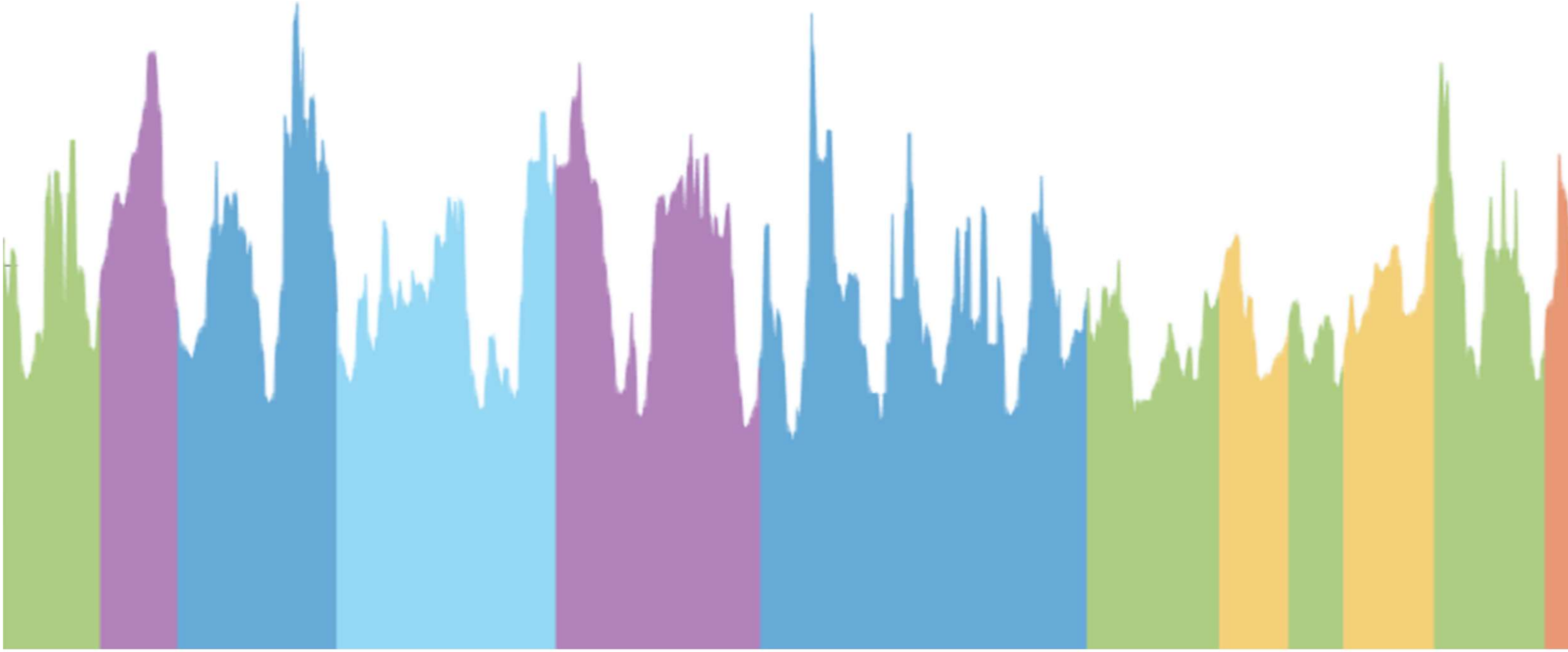
Analysis of Scale-Invariance in EEG Microstates due to Acoustic Stimuli



Master's Thesis, Rasmus Lykke

Mathematical Engineering

Aalborg University



AALBORG UNIVERSITY
STUDENT REPORT

Mathematical Sciences - Aalborg University
Frederik Bajers Vej 7B, 9220 Aalborg Øst
http://www.aau.dk

Title:

Analysis of Scale-Invariance in EEG Microstates due to Acoustic Stimuli

Theme:

Mathematical Engineering:
Master's Thesis

Project period:

September 2020 - April 2021
February 2022 - June 2022

Author:

Rasmus V. Lykke

Supervisors:

Jan Østergaard
Jesper Møller

Number of pages:

102

Date of Completion:

3$^{\text{rd}}$ of June, 2022

Synopsis:

This thesis analyzes and assesses changes in the self-similarity of EEG microstate sequences due to acoustic stimuli. By embedding the microstates into a random walk, an estimate of the Hurst exponent is obtained by means of estimation. The topic of estimating the Hurst exponent played a significant role in this thesis. Two methods were introduced for the purpose of estimating the Hurst exponent; one was an established method based on the wavelet transform; the was through the implementation of a convolutional neural network. Using the two methods, an ANOVA was performed on EEG data recordings of a listening task to assess whether the type of sound was significant. Ultimately, the results of the analysis proved inconclusive, and further work is needed on the topic.

Preface

This report was written as a 60 ECTS master's thesis by Rasmus Lykke, studying mathematical engineering at the faculty of engineering and science at Aalborg University. The project period began in the fall semester of 2020; it was temporarily suspended throughout most of 2021, before being resumed and completed in the spring semester of 2022.

I would like give a special thanks to Jan Østergaard and Jesper Møller for their guidance and support; without their feedback and advice, this project could not have been completed, and their patience with me throughout the project period is greatly appreciated. I would also like to thank two of the people who helped me during their work with the department of Electronic Systems at Aalborg University; Adèle Maryse Danièle Simon, who let me take part in an EEG recording and provided the EEG data used in the main analysis of this thesis; and Payam Shahsavari Baboukani, who helped me understand how to process EEG signals and provided me with toy datasets during the early stages of the project period.

Notation

In this report, j is used (in its non-italicized form) to denote the imaginary number $\sqrt{-1}$. $(\cdot)^T$ is used to denote transpose, $(\cdot)^*$ is used to denote complex conjugate and $(\cdot)^H$ is used to denote complex conjugate transpose.Vectors will be denoted by bold lowercase letters and matrices will be denoted by bold capital letters. Indices will be ordered as $(row, column)$ and denoted by subscript.If the name of a matrix or vector already contains a subscript (as is often the case for time-indexed vectors), this subscript will be separated from the entry indices by a semicolon when denoting entries of the matrix or vector. For example, the (i, j)th entry of a matrix $\boldsymbol{A}_k$ will be denoted $a_{k;i,j}$.

Parentheses are used to indicate the argument of functions on a continuous space (e.g. $f(t)$), while indices of a sequence will be indicated by subscript (e.g. x_n). Entries of multivariate time series are specified according to the convention established in the previous paragraph; the time index precedes the vector index, and the two are separated by a semicolon. Sets of indexed elements are denoted by brackets followed by a subscript indicating the domain of the index variable; if the order of the elements is unimportant, curly brackets are used, e.g. $\{x_i\}_{i\in\mathbb{Z}}$. If the set is a sequence of ordered elements, one of two conventions are followed; the ordering is indicated either by the use of parentheses, e.g. $(x_n)_{n\in\mathbb{Z}}$, or, if the sequence is left-bounded, by explicitly writing out the elements of the sequence in order its starting point, e.g. $x_0, x_1, \ldots$.

Whether a subscript specifies an entry of a time series or an entry of a vector should always be clear from the context, but generally, the subscript n is used to indicate time (often in $\mathbb{Z}$ or $\mathbb{N}$), while i and j, when used as subscripts, are commonly used to specify indices of vectors and matrices or to distinguish between elements of a set, in which there are no temporal characteristics.

Random variables, functions and time series will be indicated by capitalized letters. Whether a bold capitalized letter is used denote a matrix or a random vector should be clear from the context.

The second parameter of the Gaussian distribution denotes the variance, which is not to be mistaken for the standard deviation, i.e. $\mathcal{N}(\mu,\sigma^2)$ has mean μ, standard deviation σ and variance σ^2. In the context of random variables, iid. is used as shorthand for *independent and identically distributed*.

Throughout this report, the norm operator $||\cdot||$ will denote the euclidean norm, unless stated otherwise by subscript. Likewise, the term *normalized* should be taken to mean l_2-normalized, unless otherwise specified.

Contents

Introduction 1

This thesis is titled ’Analysis of scale-invariance in EEG microstates due to acoustic stimuli’. The title is sure to raise some questions for readers unfamiliar with the field of neuroscience. First, what are EEG microstates? Second, in what way do they exhibit scale-invariance?

This intro aims to answer the first question by introducing the fundamentals of EEG signals and microstates. Once the basics of the EEG microstate model have been established, the central problem of this thesis will be stated in Section 1.4.

1.1 Electroencephalography

Electroencephalography is a method for monitoring electrical activity in the brain, and has primarily been used to diagnose epilepsy and other seizure disorders. It has also been used to monitor the brain activity of comatose or sedated patients as well as patients suffering from sleep disorders.

The brain consists of billions of electrically excitable cells known as *neurons*. The neurons are able to communicate with other cells through *synapses* - structures that allow neurons to pass electrical signals to the target cell. The purpose of EEG is to monitor voltage fluctuations resulting from these electric signals.

EEG is most commonly used in a noninvasive way, by placing electrodes along the scalp in order to record cerebral electrical potentials. These potentials can appear both as action potentials, which are brief and limited in spread, and as slower and more widespread postsynaptic potentials [Binnie and Prior, 1994]. The potential recorded by a single electrode from a neural generator depends on the solid angle subtended at the electrode - meaning that electrodes placed on the scalp are too distant to record the activity of a single neuron [Binnie and Prior, 1994]. Synchronous activity in groups of neurons, however, may form a neural generator of sufficient magnitude to be recorded on the scalp [Binnie and Prior, 1994]. Thus, the EEG records the spatiotemporal averages of synchronous postsynaptic potentials across a range of cone-shaped cells.

While isolated groups of interconnected neurons adopt synchronous patterns, sensory inputs like sight and hearing stimulate individual neurons. In turn, arousal and cognitive activity reduce synchrony [Binnie and Prior, 1994]. When monitoring brain activity, patients are often at rest with their eyes closed, in which case a higher degree of synchrony is achieved.

The various frequency ranges of EEG signals correspond to different activities. The range of frequencies $0.5 - 4$ [Hz] is called the *delta band*, which shows higher activity during deep sleep [Teplan, 2002]. The range $4-7.5$ [Hz] is called the *theta band* and the range $7.5-13$ [Hz] is called the *alpha band*. The alpha band is the dominant frequency band (in terms of EEG power) in EEG of alert human adults, and is the most extensively studied EEG frequency band [Klimesch, 1999; Teplan, 2002]. Research suggests that comparatively large alpha band activity with small theta and delta band activity characterize good cognitive performance [Klimesch, 1999]. It is assumed that alpha band activity decreases and delta and theta band activity increases in the transition from alertness to sleep [Klimesch, 1999]. The range of frequencies > 13 is called the *beta band*. The beta band is dominant in alert humans with open eyes. Beta band activity significantly decreases when eyes are closed, in which case the wave pattern changes into alpha waves [Teplan, 2002].

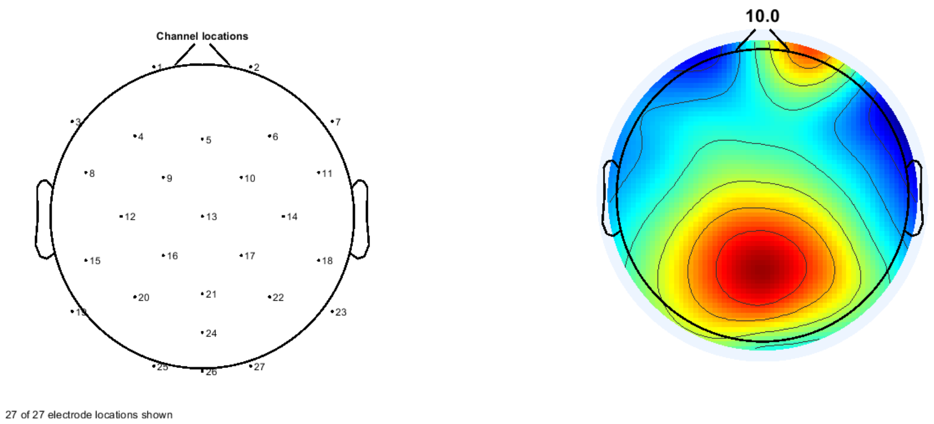
There are multiple ways to analyze EEG signals. One approach to doing so is to characterize the temporal frequency distribution of the recordings at preselected electrodes, which provides insight into the electrophysiology of the brain, but neglects to account for the multivariate characteristics of the signal [Poulsen et al., 2018, Sec. 1]. Another approach is to characterize the EEG signal by the spatial configuration of the electric fields recorded on the scalp, which can be achieved through the use of independent component analysis [Poulsen et al., 2018, Sec. 1]. The approach that will be of greatest interest to this report, however, is called *microstate analysis* and is based on topographic analysis [Poulsen et al., 2018, Sec.1].

1.1.1 EEG Topographies

The aim of this subsection is to give a brief introduction to what is meant by 'EEG topographies', by showing how EEG signals are illustrated.

EEG signals are recorded by electrodes placed on the scalp. The location of these electrodes can intuitively be communicated by means of a top-down view of the scalp. An example of a map of EEG channel locations is given in Figure 1.1a.

With each channel in the EEG signal being linked to a location on the top-down map of the scalp, each time instance of the signal can be illustrated by a topographical map Figure 1.1b, with red colors indicating high power in nearby electrodes and blue colors indicating low power in nearby electrodes. We refer to this representation of the EEG recording as an *EEG topography*.



(a) Illustration of channel locations.

(b) Topographical map of an EEG reading at a single time instance.

Figure 1.1. Examples of how EEG signals are commonly visualized.

1.1.2 Preprocessing of EEG Signals

EEG recordings may contain several unwanted elements, which need to be addressed in preprocessing.

At several points during the preprocessing, the EEG data can be average-referenced, such that the data point is a zero-sum vector. For a non-average-referenced data point $\boldsymbol{y}_n \in \mathbb{R}^{N_s}$, whose ith entry $y_{n;i}$ contains the recording at the ith electrode at time n, the average-referenced recording $\boldsymbol{x}_n \in \mathbb{R}^{N_s}$ is given as [Pascual-Marqui et al., 1995]

$$\boldsymbol{x}_n = \left(\boldsymbol{I} - \frac{1}{N_s}\mathbf{1}\mathbf{1}^\top\right)\boldsymbol{y}_n,$$

where $\boldsymbol{I}$ is the $N_s \times N_s$ identity matrix and $\mathbf{1}$ is a $N_s \times 1$ vector of ones. Throughout this thesis, when considering EEG recordings, it is assumed that they are average-referenced.

First and foremost, EEG signals contain noise, which is often modeled as a sequence of iid. Gaussian variables. Since the EEG records, each of which is prone to various errors during setup and recording. Because of this, the variance of the noise may vary between the individual channels. Channels that display extreme levels of noise relative to the other channels are referred to as *bad channels*, and are often removed from the data while preprocessing.

Noise can span the entire frequency spectrum of the signal. Since - in the context of EEG signals - different sections of the frequency spectrum correspond to different neural activities, a bandpass filter can be applied to the signal in order to remove some of the noise and to isolate the frequency bands of interest to the analysis. Additionally, if the signal were to be downsampled, applying a low-pass filter to the signal prior to the downsampling could act as a means to avoid aliasing.

Other unwanted elements in EEG signals include artifacts; patterns in the signal unrelated to brain activity that are nonetheless recorded. The causes of these artifacts are often of a physiological nature (e.g. eye movement, pulse, shivering).

A popular tool used to remove artifacts from EEG signals is *independent component analysis* (abbr. ICA). It is a computational method used to separate a multivariate signal into a set of source signals, which are assumed to be non-Gaussian and statistically independent. By identifying and removing source signals related to artifacts, the signal can be reconstructed without these artifacts.

The recorded signals are first arranged in an $N_s \times N$ matrix $\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{N-1}]$. ICA assumes that $\boldsymbol{X}$ (corresponding to an EEG channel) is a mix of $M \leq N_s$ independent source signals. In other words, ICA is a latent variable model, in which it is assumed that the observed variables are linearly related to the set of latent variables $\boldsymbol{Z} \in \mathbb{R}^{M \times N}$ as [Tharwat, 2018, Eq. (2)]

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Z},$$

where $\boldsymbol{A} \in \mathbb{R}^{N_s \times M}$ is called the *mixing matrix*. The aim of ICA is to obtain the source signals in the rows of $\boldsymbol{Z}$, but neither $\boldsymbol{Z}$ or $\boldsymbol{A}$ are known.

To obtain estimates of the source signals, it is assumed that they are non-Gaussian and - by assuming the source signals independent - factorize, so that [Bishop, 2009, Eq. (12.89)]

$$p(\boldsymbol{Z}) = \prod_{i=0}^{M-1} p_i(\boldsymbol{z}_i^\top), \tag{1.1}$$

where $p(\boldsymbol{Z})$ denotes the joint probability distribution of the source signals $\boldsymbol{z}_0^\top, \boldsymbol{z}_1^\top, \dots, \boldsymbol{z}_{M-1}^\top$, and $p_i(\boldsymbol{z}_i^\top), i = 0, 1, \dots, M-1$ denotes the marginal distributions of the indicated rows of $\boldsymbol{Z}$.

We now consider the case where the observed variables $\boldsymbol{X}$ and the source signals $\boldsymbol{Z}$ are of the same dimension, i.e. $N_s = M$. Since - in the context of EEG signals - each row of $\boldsymbol{X}$ correspond to the recording across a single electrode, this assumption implies that the recorded signal is at most composed of a number of independent components equal to the number of electrodes. We say that $\boldsymbol{X}$ is in the *electrode space* and that $\boldsymbol{Z}$ is in the *independent component space*. Assuming that the transformation from the latent variables to the observed variables is invertible, the observed variables $\boldsymbol{X}$ can be mapped to the independent component space by the transformation given as [Tharwat, 2018]

$$\boldsymbol{Z} = \boldsymbol{W}\boldsymbol{X} + \boldsymbol{V}, \tag{1.2}$$

where $\boldsymbol{W} \in \mathbb{R}^{N_S \times N_s}$ is referred to as the *unmixing matrix*. It is the inverse of $\boldsymbol{A}$, and obtaining $\boldsymbol{W}$ is the central problem of ICA. The added term $\boldsymbol{V} \in \mathbb{R}^{N_s \times N}$ is additive noise and essentially functions as an error term for the estimation. While the topic of how the mixing matrix $\boldsymbol{W}$ and source signals $\boldsymbol{Z}$ are obtained is not one of great importance to this thesis, it is worth addressing it briefly.

Some approaches to the ICA estimation problem are based on information-theoretical results. Minimizing the mutual information between the source signals $\boldsymbol{z}_i^\top, i = 0, 1, \dots, N_s - 1$ increases their independence, and entropy is larger for Gaussian variables than for any other random variable with the same variance [Cover and Thomas, 2006, Th. 8.6.5]. Thus, minimizing the mutual information and entropy should produce non-Gaussian independent source signals. A popular choice for ICA estimation is the FastICA algorithm, which seeks to maximize the negentropy of $\boldsymbol{Z}$ in (1.2) [Tharwat, 2018]. The negentropy is non-negative and zero only for for Gaussian random variables, so this approach should produce non-Gaussian source signals.

Once $\boldsymbol{W}$ and $\boldsymbol{Z}$ have been obtained, the independent components may be observed from the columns of $\boldsymbol{W}^{-1}$. By identifying which independent components are associated with different artifacts, their corresponding source signals can be set to zero. Afterwards, the original signal can be reconstructed with artifacts removed as a new matrix $\boldsymbol{X}'$ given as [Jung et al., 2000]

$$\boldsymbol{X}' = \boldsymbol{W}^{-1}\boldsymbol{Z}', \tag{1.3}$$

in which $\boldsymbol{Z}'$ is $\boldsymbol{Z}$ with the rows corresponding to artifactual components set to zero. This can be done manually; for example, by analyzing which independent components exhibit activity on electrodes near the eyes, artifactual components related to blinking or eye movement can be identified and removed.

The EEGLAB toolbox for Matlab ([Delorme and Makeig, 2004]) includes ICA decomposition functionality, using the FastICA algorithm as the default option. It allows for both manual and automatic flagging of independent components as artifacts,. For this thesis, the ICA framework included in EEGLAB will be used in the preprocessing.

One notable made apparent by (1.3) is that removing independent components makes $\boldsymbol{X}'$ rank-deficient. This may not be an issue; the information missing was presumably of an artifactual nature. Additionally, the *microstate* representation of the EEG signal, which will be introduced in the following section, assumes the existence of a sparse representation of EEG signals regardless.

1.2 EEG Microstates

Within the field of EEG analysis, *microstate analysis* is an alternative representation for EEG signals. The concept of microstates originates from research by D. Lehmann and collaborators, who observed that the topographies of EEG signals are comprised of a small discrete set of topographies that remain stable for 80-120 milliseconds before transitioning to a different topography [Lehmann, 1971; Lehmann et al., 1987]. The term *microstate* refers to these brief periods of quasi-stable topographies, while the term *microstate classes* refers to the small discrete set of topographies, which the signal cycles between. It is commonly assumed that different topographies of EEG signals reflect different configurations of neural generators. Thus, studying the dynamics of the microstates should provide insight into the changes in the neural activity of the brain.

In mathematical terms, the fundamental assumption of the microstate-representation of an EEG signal is that the (average-referenced) EEG signal at time n, $\boldsymbol{x}_n \in \mathbb{R}^{N_s}$, with N_s denoting the number of electrodes used for each measurement, can be expressed as [Pascual-Marqui et al., 1995]

$$\boldsymbol{x}_n = \sum_{i=1}^{K} a_{i,n}\boldsymbol{m}_i + \boldsymbol{v}_n, \quad n \in \mathbb{Z}, \tag{1.4}$$

where $K \in \mathbb{N}$ is the number of microstate classes, the $a_{i,n} \in \mathbb{R}$ denote the intensities of the ith microstate class at time n, the $\boldsymbol{m}_i \in \mathbb{R}^{N_s}$ are the ith microstate class template and $\boldsymbol{v}_n \in \mathbb{R}^{N_s}$ is zero-mean additive noise. Furthermore, the microstates are assumed to be non-overlapping, i.e.

$$a_{i,n}a_{j,n} = 0, \quad \forall i \neq j,$$

and it is assumed that at any time, one of the K microstate classes must be active, i.e.

$$\sum_{i=1}^{K} a_{i,n}^2 > 0, \quad \forall n.$$

Throughout this thesis, $c_n \in \{1, 2, \ldots, K\}$ will be used to denote the microstate class label at time n, i.e. the discrete number for which $a_{c_n,n} \neq 0$. The sequence $c_0, c_1, \ldots$ will be referred to as the microstate sequence. A 'microstate class' is defined by the combination of a 'microstate class label', $k \in \{1, 2, \ldots, K\}$, with its associated 'microstate class template', $\boldsymbol{m}_k \in \mathbb{R}^{N_s}$. When specifying one of the two, the former is called the and the latter is called the .

The validity of the non-overlapping assumption of microstates has been debated, with one study by Mishra et al. [2020] arguing that microstates are not as discretely isolated as is commonly assumed, and that the assumption of discreteness is less valid for points of the EEG recording with low power and low signal-to-noise ratio. Nonetheless, the assumption is commonplace in the relevant litterature.

An analysis of microstate sequences typically involves analyzing how the dynamics of microstates change under different experimental conditions. The dynamics of each microstate class include their mean duration, their frequency of occurence and the percentage of analysis time occupied by them. Examples include Koenig et al. [2002], in which the interaction between microstate dynamics and age was investigated in order to gain a better understanding of the maturation of the brain; their findings suggested a complex evolution of microstate dynamics with age, compatible with brain developmental stages proposed by developmental psychologists.

Another approach to analyzing microstate sequences is to estimate the transition probabilities between different microstate classes and analyze how they change under different experimental conditions. The estimated transition probabilities typically take the form of a transition matrix, in which the entry $p_{i,j}$ is the probability that the microstate sequence transitions from microstate class i to microstate class j. Al Zoubi et al. [2019], for example, analyzed how the dynamics of microstate sequences differed for individuals with mood and anxiety disorders versus healthy control subjects. They found that while the microstate class topographies and microstate dynamics showed little to no significant difference between the two groups, some transition probabilities showed significant differences between the two groups. However, in estimating the transition probabilities between microstate classes as a transition matrix, it is implicitly assumed that the microstate sequence is a first order Markov chain (as defined later in Definition A.28), and that the probabilities are stationary and do not change over time.

von Wegner et al. [2017] tested microstate sequences for the Markov property of orders 0, 1 and 2, corresponding to the three hypotheses $P(c_n|c_{n-1}, c_{n-2}, \dots) = P(c_n)$, $P(c_n|c_{n-1}, c_{n-2}, \dots) = P(c_n|c_{n-1})$ and $P(c_n|c_{n-1}, c_{n-2}, \dots) = P(c_n|c_{n-1}, c_{n-2})$, respectively. They found that the microstate sequences fulfilled none of the three properties, concluding that the memory effect of microstate transitions extends further than two steps into the past. Additionally, when testing for stationarity of the transition matrix (whether the transition probabilities are independent of time), the results suggested that the transition matrix was non-stationary and change over time. Finally, when testing for symmetry of the transition matrix (the hypothesis that $P(c_n = j|c_{n-1} = i) = P(c_n = i|c_{n-1} = j)$), they found that the transition matrix was asymmetric in a majority of cases. The study by Al Zoubi et al. [2019] used earlier in this section as an example also applied the methodology of von Wegner et al. [2017] to their data, obtaining similar findings. This suggests that microstate sequences are not low-order Markovian, have neither time-invariant nor symmetric transition probabilities and may exhibit long-term memory properties.

Identifying and classifying the sequences of microstates is a post-hoc process. The microstate classes must first be identified before each point of an EEG signal can be assigned a microstate class. Clustering algorithms are commonly used for this task. A popular example of such an algorithm is the *K-means* algorithm, which iteratively assigns each point to the cluster whose mean is the shortest distance from the point, after which it updates the mean of each cluster based on the points assigned to it. Another example is a polarity-invariant version of the K-means algorithm called the *modified K-means* or *N-microstates* algorithm [Pascual-Marqui et al., 1995]. The assumption (1.4) is consistent with this algorithm. Both algorithms will be introduced in Section 1.3.

While they won't be considered for this thesis, popular alternatives to K-means and modified K-means for microstate estimation do exist. An example is the *Topographic Atomize and Agglomerate Hierarchical Clustering*, which is a hierarchical clustering method. Unlike K-means, it does not require a preset number of clusters. The algorithm starts with each point being assigned its own cluster and iteratively removing the cluster with the lowest sum of correlations between it and its members, before reassigning the points in the removed cluster to their most similar cluster [Poulsen et al., 2018]. Probabilistic alternatives such as the *fuzzy C-means* have also been suggested [Dinov and Leech, 2017], since these allow for a single topographical map be assigned a 'degree of membership' for each microstate class, as opposed to being assigned only one microstate class. As such, the fuzzy C-means algorithm agrees more with analysises that reject the non-overlapping assumption of the microstate model.

In order to avoid using noisy measurements when estimating the prototype microstates, it is common practice to only use the data points which correspond to the maximal signal-to-noise ratio. For EEG data, these points are generally found at the peaks in the *global field power* (abbr. GFP) [Dinov and Leech, 2017], which is defined as follows:

Definition 1.1 (Global Field Power)
For an average-referenced measurement $\boldsymbol{x}_n = \left[x_{n;0}, x_{n;1}, \dots, x_{n;N_S-1}\right]^\top$, where N_s is the number of electrodes used for the measurement, the GFP is defined as [Murray et al., 2008]

$$\mathrm{GFP}_n = \sqrt{\frac{1}{N_s} \cdot \sum_{i=0}^{N_s-1} x_{n;i}^2}. \tag{1.5}$$

▲

Mathematically, the GFP is identical to taking the root mean square across all channels of an average-referenced EEG reading at a given time instance.

After having found the peaks of the GFP and used them to estimate the microstate class prototypes, the microstate classes must be 'backfitted' onto the original signal. A measure often used for assigning data points to microstate classes is *global map dissimilarity* (abbr. GMD) [Poulsen et al., 2018, Eq. (18)]. It is a distance measure, and can be defined as follows:

Definition 1.2 (Global Map Dissimilarity)
For two EEG samples $\boldsymbol{x}_m, \boldsymbol{x}_n \in \mathbb{R}^{N_s}$, the global map dissimilarity is defined as [Poulsen et al., 2018, Eq. (18)]

$$\mathrm{GMD}_n = \frac{\left\| \frac{\boldsymbol{x}_m}{\mathrm{GFP}_m} - \frac{\boldsymbol{x}_n}{\mathrm{GFP}_n} \right\|}{\sqrt{c}}, \tag{1.6}$$

where GFP_n is the GFP of the signal at time n and c is a normalizing constant. ▲

1.2.1 Canonical EEG Microstates

In resting state subjects with eyes closed, four specific microstate topographies appear consistently (e.g. [Van De Ville et al., 2010; Al Zoubi et al., 2019]), and in [Koenig et al., 2002], a study of 496 subjects, they were found to account for 79% of the variance in the data on average. Since microstates are hypothesized to be a set of 'building blocks' for brain electric activity, each incorporating different modes or steps of information processing [Lehmann et al., 1998], these four microstates, shown in Figure 1.2, are likely to play a significant role in human thought. Hence, they are often called the *canonical microstates*.

While Figure 1.2 shows the four canonical microstates obtained from real EEG data, there is bound to be some difference due to randomness in the data.Additionally, since the clustering methods used for microstate class estimation do not assume any prior knowledge of the four canonical microstates, the order in which the microstates are sorted is not consistent and labeling estimated microstate classes as one of the canonical microstates is a manual process. Figure 1.2 shows this issue; while the classes have been rearranged to match the litterature, the microstate classes were originally sorted according to frequency of occurrence (indicated by the number above each class). To ensure a consistent characterization of the four canonical microstates, *A*, *B*, *C* and *D*, they can be defined by their topographical characteristics as follows:

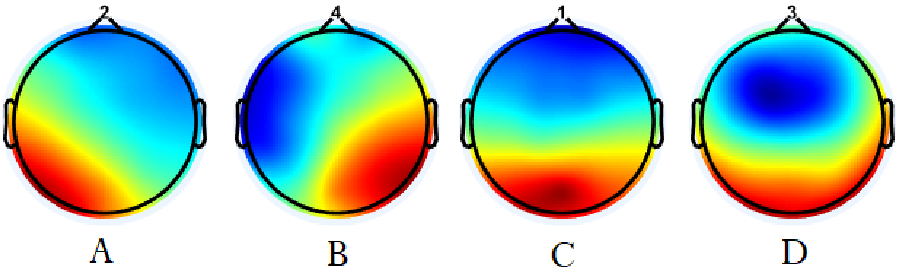


Figure 1.2. The four microstate classes obtained from EEG data using EEGLAB. The top numbers indicate the order of frequency of appearance, while the bottom letters indicate the labels used in the relevant litterature [Koenig et al., 1999; Britz et al., 2010].

(A) Right anterior-left posterior orientation.

(B) Left anterior-right posterior orientation.

(C) Anterior-posterior orientation.

(D) Central extremum.

1.3 Clustering Algorithms

Microstates are defined using *clustering algorithms*. One of the simplest ways to interpret clusters is to think of them as groups of multidimensional data points in which the distance between two points within the same group is small compared to the distance between two points not within the same group. The aim of a clustering algorithm is to identify clusters of data points in a multidimensional space. K-means is one such algorithm and aims to separate points from a set of data into K clusters, each of which has a mean (also referred to as the cluster center). Each point in the dataset is assigned to the cluster with the nearest mean. The aim of this section is to introduce the K-means algorithm as well as alternatives to the algorithm; a modified variant of the K-means algorithm introduced by [Pascual-Marqui et al., 1995] - dubbed the *N-microstates* algorithm by the authors.

1.3.1 *K*-Means

Unless otherwise stated, this section is based on [Bishop, 2009, Sec. 9.1].

We start by considering how a set of N_T N_s-dimensional data points $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_T-1}\} \subset \mathbb{R}^{N_s}$ can be individually classified as belonging to one of K different clusters. The K-means algorithm aims to characterize each of K clusters by their own mean vector, $\boldsymbol{\mu}_k \in \mathbb{R}^{N_s}$, hence the name of the algorithm. In the context of EEG, the size N_s corresponds to the number of electrodes used for the recording, N_T corresponds to the number of temporal samples used and K corresponds to the number of microstates considered in the model.

The initial goal of the K-means is to find the assignment of data points to clusters and the set of vectors $\mu_0, \mu_1, \ldots, \mu_{K-1}$ that minimizes the sum of squares of the distance between each data point and its closest vector μ_k. The reasons as to why the set of vectors $\mu_0, \mu_1, \ldots, \mu_{K-1}$ can be thought of as mean vectors will become apparent soon. For now, this initial goal can be stated as the optimization problem [Bishop, 2009, Eq. (9.1)]

$$\underset{\{\boldsymbol{\mu}_k\},\{r_{k,n}\}}{\text{minimize}} \quad \sum_{n=0}^{N_T-1} \sum_{k=0}^{K-1} r_{k,n} \left\| \boldsymbol{x}_n - \boldsymbol{\mu}_k \right\|^2 \quad , \tag{1.7}$$

where $r_{k,n}$ is 1 if $\boldsymbol{x}_n$ is assigned to cluster k and 0 otherwise.

In the K-means algorithm, the objective function in (1.7) is minimized by an iterative procedure, in which we alternate between minimizing the objective function with respect to $\{r_{k,n}\}$ while keeping $\{\boldsymbol{\mu}_k\}$ fixed, and minimizing the objective function with respect to $\{\boldsymbol{\mu}_k\}$ while keeping $\{r_{k,n}\}$ fixed.

Using superscript to denote iteration number, we start by choosing some initial values for the set of vectors $\left\{\mu_0^{(0)}, \mu_1^{(0)}, \ldots, \mu_{K-1}^{(0)}\right\}$. In practice, this can be achieved by use of a pseudo-random number generator.

We now keep $\left\{\mu_0^{(0)}, \mu_1^{(0)}, \ldots, \mu_{K-1}^{(0)}\right\}$ fixed and seek to minimize the objective function in (1.7) with respect to $\{r_{k,n}\}$. By utilizing the fact that the objective function in (1.7) is a linear function of $r_{k,n}$ while noting that the terms involving different n are independent, we can optimize for each n separately, by simply letting $r_{k,n}$ be 1 for whichever value of k yields the minimum value of $\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$. This step can be written as [Bishop, 2009, Eq. (9.2)]

$$r_{k,n}^{(i+1)} = \begin{cases} 1 & \text{if} \quad k = \underset{j}{\text{argmin}} \ \|\boldsymbol{x}_n - \boldsymbol{\mu}_j^{(i)}\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{1.8}$$

Next, keeping $r_{k,n}$ fixed, we seek to minimize the objective function in (1.7) with respect to $\left\{\mu_0^{(0)}, \mu_1^{(0)}, \ldots, \mu_{K-1}^{(0)}\right\}$. We consider this optimization separately for each $\boldsymbol{\mu}_k$. This turns the optimization problem into that of a quadratic program, meaning that a minimum solution can be obtained on closed form by setting the derivative of the objective function with respect to $\boldsymbol{\mu}_k$ to zero and solving for $\boldsymbol{\mu}_k$, yielding [Bishop, 2009, Eq. (9.3)]

$$\boldsymbol{\mu}_k^{(i+1)} = \frac{\sum_{n=0}^{N_T-1} r_{k,n}^{(i+1)} \boldsymbol{x}_n}{\sum_{n=0}^{N_T-1} r_{k,n}^{(i+1)}}. \tag{1.9}$$

Since the numerator in the sum of all data points in cluster k and the denominator counts the number of data points in cluster k, $\boldsymbol{\mu}_k^{(i)}$ is the mean of all data points assigned to cluster k at the ith iteration.

While the K-means is guaranteed to converge to a minimum since the value of the objective function is reduced at each iteration, it may converge to a local minimum rather than a global one. Nonetheless, the K-means algorithm has been popular in computing EEG microstates.

An example of microstates obtained from EEG data using K-means clustering with different values of K is shown in Figure 1.3. One thing to note in this figure is that the K-means algorithm is not polarity invariant; topographical maps that are (approximately) identical up to a

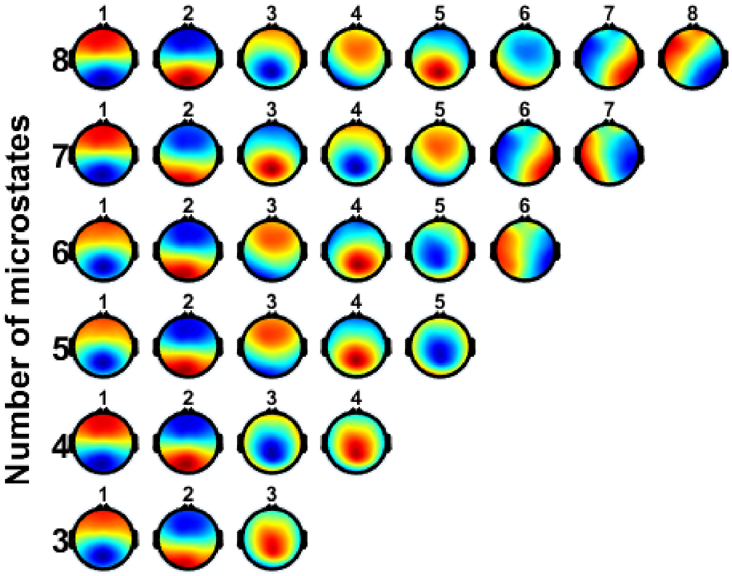


Figure 1.3. Topographical view of examples of microstates obtained from EEG data using K-means with K=3,4,...,8 as indicated by each row. The microstates were computed using the Microstate toolbox ([Poulsen et al., 2018]) for EEGLAB ([Delorme and Makeig, 2004]).

negative scalar are assigned to different clusters by the K-means algorithm. This means that microstates that are virtually identical in terms of their topography are treated as being unrelated to one another. This issue is addressed in the modified K-means algorithm.

1.3.2 Modified K-Means

The aim of this subsection is to introduce the modified K-means or N-microstates algorithm of [Pascual-Marqui et al., 1995], which is similar to K-means in multiple ways. Throughout the rest of this thesis, it will be referred to by the shortened name 'mod-K-means'. In order to avoid excessive redundancy, this subsection serves to only briefly introduce the algorithm and highlight how it differs from the regular K-means.

Similarly to K-means, the aim of mod-K-means is to assign each data points to one of K clusters, each characterized by their own vector, $\boldsymbol{\gamma}_k$. Note that the number of clusters is still denoted by K, but the characteristic vector is now denoted by $\boldsymbol{\gamma}_k$ rather than $\boldsymbol{\mu}_k$, as to not confuse it with a mean vector.

The mod-K-means algorithm is specifically tailored for EEG signals, so the data point $\boldsymbol{x}_t \in \mathbb{R}^{N_s}$ can be thought of as containing the scalp potential measurements at time n. N_s denotes the number of electrodes used for the EEG recording.

For the K-microstates algorithm, the optimization problem is given as [Pascual-Marqui et al., 1995, Eq. (2),(4)]

$$\begin{array}{ll} \underset{\{\boldsymbol{\gamma}_k\},\{a_{k,n}\}}{\text{minimize}} & \frac{1}{N_T(N_s-1)}\sum_{t=0}^{N_T-1}\left\|\boldsymbol{x}_n-\sum_{k=0}^{K-1}a_{k,n}\boldsymbol{\gamma}_k\right\|^2, \\ \text{subject to} & a_{k,n}\cdot a_{j,n}=0, \forall k\neq j, \\ & \sum_{k=0}^{K-1}a_{k,n}^2 \quad \geq 0, \forall n, \end{array} \quad . \tag{1.10}$$

At first, the $a_{k,n}$ appear to be very similar to the $r_{k,n}$ from K-means, since the first constraint ensures that for any given t, $a_{k,n}\neq 0$ for only one value of k. This in turn means that the sum over k in the objective function in (1.10) reduces to a single term at each n, in which case the objective functions of (1.7) and (1.10) are nearly identical up to a constant scalar. One notable difference between the two, however, is that the value of the nonzero $a_{k,n}$ need not be 1. Instead, $a_{k,n}$ is a measure of the kth microstate intensity at time instant n. In turn, this means that the method of obtaining solutions for this optimization problem differs from that of K-means.

We start by choosing a normalized and linearly independent set of initial vectors $\left\{\boldsymbol{\gamma}_0^{(0)},\boldsymbol{\gamma}_1^{(0)},\ldots,\boldsymbol{\gamma}_{K-1}^{(0)}\right\}$ and seek to minimize the objective function of (1.10) with respect to $a_{k,n}$ under the given constraints, while keeping $\boldsymbol{\gamma}_k$ fixed. To do so, we use L_n to denote the microstate label associated with the data point $\boldsymbol{x}_n$ and label each data point as belonging to the microstate to which it is closest in terms of the orthogonal squared distance. This can be expressed as follows [Pascual-Marqui et al., 1995, Eq. (5)]

$$L_n^{(i+1)} = \underset{k}{\text{argmin}}\ \boldsymbol{x}_n^H\boldsymbol{x}_n-\left(\boldsymbol{x}_n^H\boldsymbol{\gamma}_k^{(i)}\right)^2.$$

The microstate intensity $a_{k,n}$ can now be estimated as [Pascual-Marqui et al., 1995, Eq. (7)]

$$a_{k,n}^{(i+1)}=\begin{cases}\boldsymbol{x}_n^H\boldsymbol{\gamma}_k^{(i)} & \text{if} \quad k=L_n^{(i+1)}\\ 0 & \text{otherwise.}\end{cases} \tag{1.11}$$

Next, we seek to minimize the objective function of (1.10) with respect to $\boldsymbol{\gamma}_k$, while keeping the labels L_n and microstate intensities $a_{k,n}$ fixed. The minimum of the objective function is obtained as the normalized eigenvector corresponding to the largest eigenvalue of the matrix given as [Pascual-Marqui et al., 1995, Eq. (8)]

$$\boldsymbol{S}_k^{(i+1)}=\sum_{\left\{n:L_n^{(i+1)}=k\right\}}\boldsymbol{x}_n\boldsymbol{x}_n^H.$$

The estimate of $\boldsymbol{\gamma}_k^{(i+1)}$ can be obtained as [Pascual-Marqui et al., 1995, Eq. (9)]

$$\boldsymbol{\gamma}_k^{(i+1)}=\underset{\{\boldsymbol{e}:\|\boldsymbol{e}\|=1\}}{\text{argmax}}\ \boldsymbol{e}^H\boldsymbol{S}_k^{(i+1)}\boldsymbol{e} \tag{1.12}$$

An example of microstates obtained from EEG data using the mod-K-means with various choices of K is shown in Figure 1.4. In this figure, it is clear that the mod-K-means topographies are polarity-invariant (i.e. proportional but opposite topographies are assigned to the same cluster), as opposed to those obtained by use of K-means in Figure 1.3.

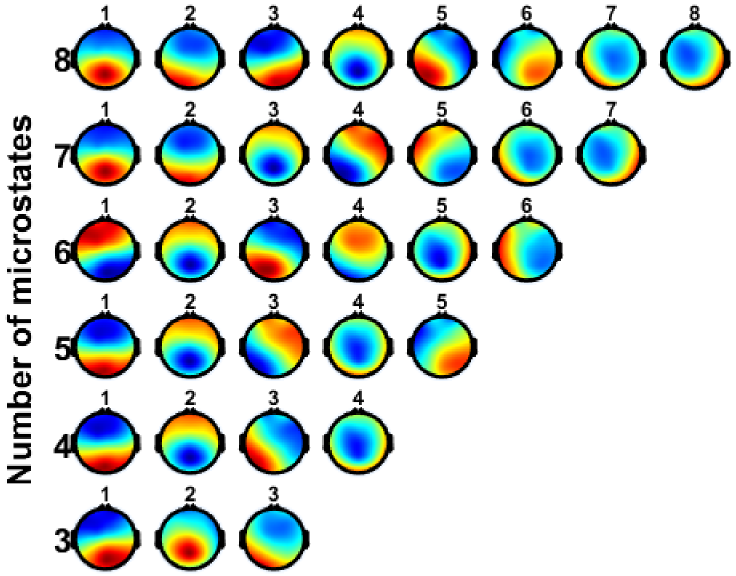


Figure 1.4. Topographical view of examples of microstates obtained from EEG data using mod-K-means with K=3,4,...,8 as indicated by each row. The microstates were computed using the Microstate toolbox ([Poulsen et al., 2018]) for EEGLAB ([Delorme and Makeig, 2004]).

1.4 Problem Statement

In recent years, the microstate representation of EEG signals has received significant attention. The reason behind its popularity is two-fold: it has been hypothesized that microstate dynamics reflect changes in neural activity, and the microstate representations of EEG signals provide a significant dimensionality reduction, since every point in time takes a discrete value from a finite set rather than a vector of continuous variables.

Studies have indicated that microstate sequences exhibit fractal properties [Van De Ville et al., 2010]; specifically, temporally scaled versions of the microstate sequences have been found to be statistically similar across a range of scales, suggesting that neural activity occurs across multiple temporal scales. This implies that microstates could exhibit some form of *scale-invariance* - being statistically identical under any rescaling in time. However, since the processed EEG signals are discrete in time and finite in length, the hypothesized scale-invariance must be investigated by using a less restrictive fractal property related to scale-invariance as proxy. This related property is *self-similarity*, which is invariant to a discrete set of scalings; it can be characterized by the Hurst exponent.

The nature of the self-similar behavior of microstate sequences is central to this thesis. Since changes in microstate dynamics are assumed to reflect changes in neural activity, a question could be raised as to which characteristics of neural behavior are revealed from changes in the Hurst exponent, which is frequently used in assessing long-range dependence of a signal.

One could speculate whether low levels of long-range dependence suggest that the neural circuit operates in a rapid and volatile manner, indicating some level of distress or arousal in the subject. The most direct approach to affecting the mental state of the subject is by introducing external stimuli - with a natural choice of stimuli being those of an acoustic nature. Under various types of acoustic stimuli, changes in the Hurst exponent can be observed, potentially revealing new information about the dynamics of the brain. Ultimately, the neuroscientific interest in this issue is largely tangential to the field of engineering mathematics; therefore, this thesis adopts an exclusively model-driven approach to the issue instead. This discussion motivates the following statement, which, from here on, will act as the central problem of the thesis:

This thesis investigates changes in the self-similar behavior of EEG microstates due to acoustic stimuli.

The acoustic stimuli in this thesis (introduced later in Chapter 4) includes music and speech, specifically, each of which has two subgroups. The music signals include piano pieces or pieces of an electronic nature; neither contain vocals. The speech signals are obtained from either a male or a female speaker. The subject was presented with an attention task, in which they were asked to focus on one of two concomittant sounds played on two separate loudspeakers, during which their scalp potentials were recorded with an EEG.

The task stated above composed of multiple steps, as depicted in Figures 1.5 and 1.6. Figure 1.5 provides a general overview of the process, emphasizing the steps that have already been discussed in this section. Figure 1.6 expands upon the fractal analysis segment of Figure 1.5, which will be the main concern of this thesis moving forward.

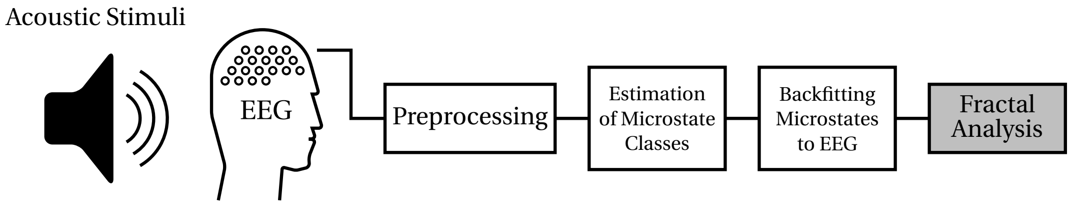

Acoustic Stimuli
EEG
Preprocessing
Estimation of Microstate Classes
Backfitting Microstates to EEG
Fractal Analysis

Figure 1.5. The system diagram of the central problem of this thesis.

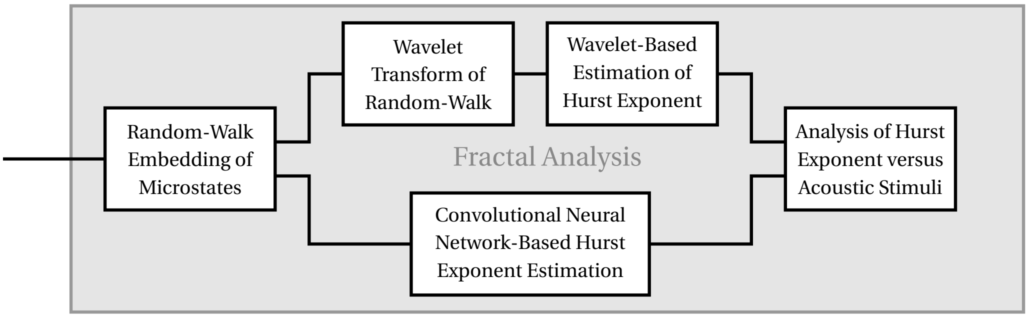

Random-Walk Embedding of Microstates
Wavelet Transform of Random-Walk
Wavelet-Based Estimation of Hurst Exponent
Fractal Analysis
Convolutional Neural Network-Based Hurst Exponent Estimation
Analysis of Hurst Exponent versus Acoustic Stimuli

Figure 1.6. The system diagram for the fractal analysis.

While the fundamentals of EEG recordings were introduced in the beginning of this chapter, the problem of recording EEG is not central to this thesis.

The *Preprocessing* step concerns the steps introduced in Section 1.1.2, including bandpass-filtering and removal of artifacts using ICA.

Estimation of Microstates is a two-step process; first, the peaks in the GFP (Definition 1.1) are identified. Next, microstate classes are estimated based on the data points corresponding to peaks in the GFP using mod-K-means (Section 1.3.2).

The *Backfitting of Microstates to EEG* is the process of assigning each point of the original signal a specific microstate class.

The last step, *Fractal Analysis*, includes the substeps in Figure 1.6, the majority of which will be introduced and discussed in Chapter 2. The first step of the fractal analysis is transforming the EEG microstate sequences into a series suitable for fractal analysis using a technique referred to as *random-walk embedding*. This will be discussed in Section 2.3.

The second step is to assess the self-similarity of the transformed series of EEG microstates from a Hurst exponent analysis, either by application of an existing wavelet-based analysis or by a convolutional neural network-based approach. The concept of statistical self-similarity will be introduced in Section 2.2. While general theory of wavelets has been relegated to Appendix B, the wavelet-based method, by which self-similarity of the microstate sequences can be assessed, will be introduced in Section 2.4. Similarly, general theory of deep learning and artificial neural networks can be found in Appendix C, while the topic of a deep learning-based analysis of self-similarity will the main focus of Chapter 3.

The final step includes the experimental aspect of this thesis; the analysis of whether changes occur in the Hurst exponents for different types of acoustic stimuli. Chapter 4

The field of EEG microstate estimation and classification has been extensively studied, and there are tools available for easy application of microstate estimation and classification [Poulsen et al., 2018]. Moving forward, the process of microstate estimation and classification will not receive much attention. Instead, the focus will be on the fractal analysis, which - to the extent of the author's knowledge - is less extensively researched and more open to innovation.

Specifically, this thesis aims to address the following points:

- Application of a convolutional neural network-based approach to assessing fractal properties of microstate sequences.
- Assessment of changes in the Hurst exponent of microstate sequences in subjects exposed to acoustic stimuli.

Fractals and Statistical Self-Similarity 2

In [Van De Ville et al., 2010], it was proposed that EEG microstate sequences exhibit scale-free dynamics and statistical self-similarity. The purpose of this chapter is to introduce the fundamental theory of fractal theory with an emphasis on the concept of statistical self-similarity and discuss how the degree of self-similarity of a stochastic process can be assessed.

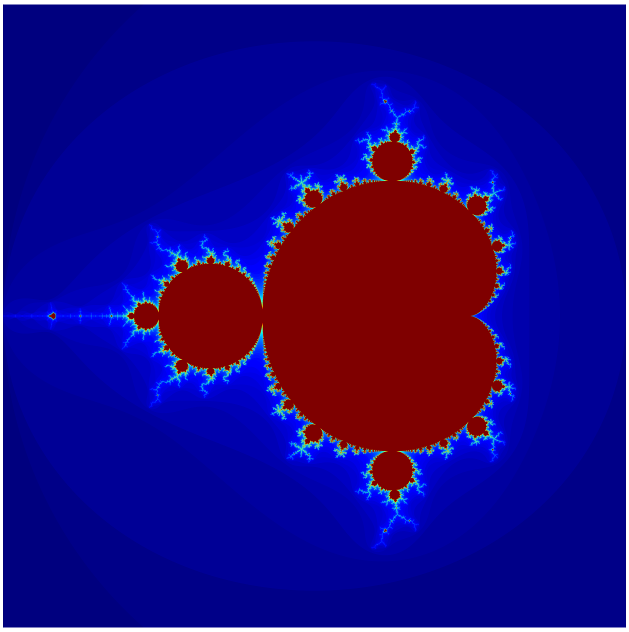


Figure 2.1. A popular example often used in fractal theory is the Mandelbrot set, the boundary of which is a fractal curve.

2.1 Fractals

The aim of this section is to introduce the definition of a fractal. Informally, a *fractal* is a subset of a topological space which appears to be the same regardless of the scale at which it is viewed. The property of similar patterns appearing at different scales is referred to as *self-similarity*, and will be the main topic of Section 2.2. While fractals can exist in any topological space, the theory introduced in this section will be restricted to fractals on $\mathbb{R}^n$. In order to provide context, important observations from the early works on fractals by mathematician Benoit Mandelbrot and the concept of a non-integer dimension will first be introduced. Later, in Section 2.1.2, a formal definition of fractals will be introduced.

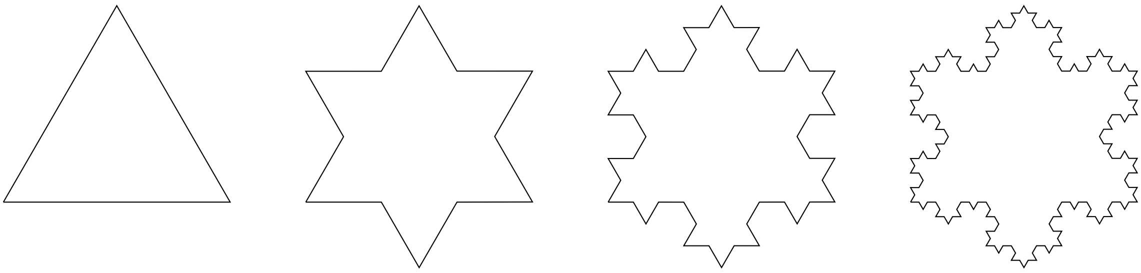


Figure 2.2. The first three iterations of the Koch snowflake.

Example 2.1 (Koch Snowflake)
Consider the first three iterations of the Koch snowflake illustrated in Figure 2.2. Starting with a simple equilateral triangle, the Koch snowflake is obtained by iteratively adding a smaller equilateral triangle to each side of the shape. The sides of these smaller equilateral triangles are one third the length of those of the previous iteration. At each iteration, the number of triangles added increases by a factor of four, while the area of each of the added triangles decreases by a factor of 9. Assuming that the first triangle has unit area, the area A of the Koch snowflake can be expressed as the geometric series

$$A = 1 + \sum_{i=0}^{\infty} \frac{1}{3} \left(\frac{4}{9}\right)^i .$$

This is an infinite geometric series, which converges to $1 + \frac{\frac{1}{3}}{1-\frac{4}{9}} = \frac{8}{5}$. The perimeter of the Koch snowflake, on the other hand, increases by a factor of 4/3 at each iteration. This means that the Koch snowflake is a geometric shape of finite area and infinite perimeter. ▲

2.1.1 Fractional Dimensions

The term ’fractal’ originates from Mandelbrot [1967], which considered the issue of measuring the length of the coastline of Britain by approximating it as the perimeter of a polygon with sides of equal length, whose corners lie on the coastline. Decreasing the length of each side leads to finer details of the coastline being taken into account and causes the approximated length to increase. Since the concept of length is undefinable for geographical curves, Mandelbrot [1967] argued that the traditional concept of dimension should be expanded to allow for fractional dimensions when dealing with ’rough’ curves such as the British coastline.

To provide an intuitive interpretation of a fractional dimension, recall that the perimeter of a polygon is simply given as the sum of the lengths of each side. This can be seen as raising the lengths to the power 1, which is the Euclidean dimension of a straight line [Mandelbrot, 1982, Ch. 5]. Similarly, the surface area of a closed polygon is calculated by covering the surface of the polygon with squares and adding together the sides of the squares raised to the power 2, which is the Euclidean dimension of a plane [Mandelbrot, 1982, Ch. 5]. In general, we have that a length of size ϵ results in a measure (e.g. the linear measure or the surface area) of size ϵ^D, where D is the dimension. The measure only has a meaningful interpretation when the correct value of D is considered, which will be elaborated upon below.

Consider now a polygonal approximation of a complicated geometrical curve (for example, a coastline) by means of several connected line segments, each of which has length ϵ. The most intuitive way to approximate the length of the geometrical curve is to multiply the number of line segments, N, by the length of each line segment, ϵ, in which case we implicitly assume that we have the dimension $D = 1$. In order to generalize this approach to be applicable to any choice of dimension, one may choose to use the estimate $L(\epsilon) = N\epsilon^d$ instead, where d denotes the dimension chosen for the estimation, as opposed to the true dimension D. When estimating a curve by use of connected lines, the number of line segments used for such an approximation has been found to be [Mandelbrot, 1982, Ch. 5]

$$N = F\epsilon^{-D}, \tag{2.1}$$

where F is some scalar related to the size of the geometric curve and D is the true dimension. Intuitively, this makes sense for simple geometric curves where $D = 1$, since the number of line segments should be inversely proportional to the length of each individual segment.

When estimating geometric curves, the dimension d is conventionally considered to be integer-valued (as is the case for the Euclidean dimension). However, this assumption may result in the measure tending to ∞ for $\epsilon \to 0$ when dealing with rough geometric curves (for example, the coastline mentioned earlier). To circumvent this issue, we allow the dimension d to be any positive value and assume that the true dimension D can be any positive value as well. For the linear measure, we have from (2.1) that $L(\epsilon) = N\epsilon^d = F\epsilon^{d-D}$. We see that [Mandelbrot, 1982, Ch. 5]

$$\lim_{\epsilon \to 0} L(\epsilon) = \begin{cases} F, & d = D \\ \infty, & d < D \\ 0, & d > D. \end{cases} \tag{2.2}$$

This indicates that the approximate measure $L(\epsilon)$ behaves reasonably if and only if $d = D$, in which case, the approximate measure is independent of ϵ [Mandelbrot, 1982, Ch. 5]. The discussion of the linear measure $L(\epsilon)$ in this section is the intuitive motivation behind the *Hausdorff dimension*, which will be introduced in the next subsection.

2.1.2 Fractal Definition

In [Mandelbrot, 1982], Mandelbrot defined a fractal as "a set for which the Hausdorff-Besicovitch dimension strictly exceeds its topological dimension." For fractals on $\mathbb{R}^n$, the topological dimension is simply the Euclidean dimension.

In order to obtain a formal definition of the Hausdorff-Besicovitch dimension, we first introduce a definition for the *Hausdorff measure* [Hutchinson, 1981, Sec. 2.6]:

Definition 2.1 (Hausdorff Measure)

Let $d \geq 0$ be a fixed real number. For every $r > 0$ and $S \subset \mathbb{R}^n$, we define

$$H_r^d(S) = \inf\left\{\sum_{i=1}^{\infty} \alpha_d 2^{-d} \left(\operatorname{diam}(S_i)\right)^d : S \subset \bigcup_{i=1}^{\infty} S_i, \operatorname{diam}(S_i) \leq r\right\}, \tag{2.3}$$

in which d is used as superscript and α_d is a normalizing constant and $\operatorname{diam}(\cdot)$ is the diameter of a set (Definition A.3). The d-dimensional *Hausdorff measure* is given as

$$H^d(S) = \lim_{r \to 0} H_r^d(S) \tag{2.4}$$

▲

The Hausdorff measure is related to the approximated linear measure $L(\epsilon)$ from the previous subsection. In fact, (2.3) can be considered a generalization of the approximated linear measure $L(\epsilon)$. Using this measure, we can define the Hausdorff-Besicovitch dimension of a set [Hutchinson, 1981, Sec. 2.6]:

Definition 2.2 (Hausdorff-Besicovitch Dimension)

The Hausdorf-Besicovitch dimension $\dim_H(S)$ of a set $S \subset \mathbb{R}^n$, is the smallest value of d for which the d-dimensional Hausdorff measure (Definition 2.1) is zero. In other words,

$$\dim_H(S) = \inf\left\{d \geq 0 : H^d(S) = 0\right\}.$$

▲

Informally, the Hausdorff-Besicovitch dimension can be interpreted as a measure of the *roughness* of a set. While it agrees with the Euclidean dimension for smooth shapes such as line segments, squares or cubes, it exceeds the Euclidean dimension for sets which exhibit roughness, for example the boundaries of the Mandelbrot set Figure 2.1 and Koch snowflake Figure 2.2. The latter example will be introduced in Example 2.2.

With the Hausdorff-Besicovitch dimension defined, Mandelbrot's definition of a fractal can be restated in a formal manner.

Definition 2.3 (Fractal)

A *fractal* is a set (in $\mathbb{R}^n$) for which the Hausdorff-Besicovitch dimension (Definition 2.2) strictly exceeds its Euclidean dimension [Mandelbrot, 1982, Ch. 3]. ▲

To the extent of the author's knowledge, there is no straightforward method for determining the Hausdorff-Besicovitch dimension of an arbitrary set. Nonetheless, the following example provides an intuitive approach to assessing the Hausdorff-Besicovitch dimension of a geometric curve.

Example 2.2 (Koch Curve)

Consider the Koch curve in Figure 2.3, which depicts a section of the Koch snowflake depicted in Figure 2.2. The Koch curve in this figure is obtained using a straight horizontal line as the initiator for the Koch curve. From Figure 2.3, we see that, by scaling the Koch curve on the left by a factor 3, we obtain the segment on the right. Due to the self-similarity of the Koch curve, the segment on the right can be constructed as the original segment repeated four times. Thus,

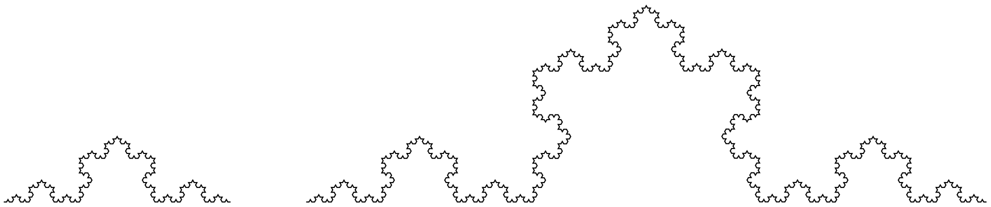


Figure 2.3. Scaling a Koch curve by a factor 3 increases its length by a factor 4.

the length of the scaled segment is four times that of the original segment. By letting D denote the dimension of the curve, we have $3^D = 4$, and we have that $D = \log_3 4 \approx 1.26$. This number is larger than the Euclidean dimension of the curve, which is 1 [Mandelbrot, 1982, Ch. 6]. By Definition 2.3, this makes the Koch curve a fractal. ▲

The example above leveraged the fact that the scaled version of the Koch curve consists of repeats of the original version in order to infer the Hausdorff-Besicovitch dimension. This property is what we refer to as *self-similarity*. While not all fractals exhibit strict self-similarity, it is one of the most significant features associated with fractals, especially in the context of microstate sequences.

2.2 Self-Similarity

While the terms *self-similarity* and *scale-invariance* have both been addressed in a rather superficial manner earlier in this thesis, no formal definition have been provided for either of the two terms.

’Scale-invariance’ is the more restrictive of the two properties, since it implies that the scale-invariant object is identical to its scaled counterpart for any scaling. Homogenous functions are an example of scale-invariant functions, since $f(st) = s^d f(t)$ for a homogeneous function of degree d. The fractals used as examples in this chapter are not scale-invariant, and since the recordings used are ultimately discrete in time, the self-similarity property, which requires the fractal to be invariant to only a discrete set of scalings, is more relevant to this thesis.

The aim of this section is to provide a definition of *self-similarity*. which is a fundamental feature that characterizes most fractals. Self-similar fractals are fractals for which a subset of the fractal, when scaled to the size of the whole, is indistinguishable from the whole [Mandelbrot, 1967]. This feature was leveraged in Example 2.2, in which the right figure of Figure 2.3 was identical to the left figure repeated four times. Formally, self-similarity is defined as follows:

Definition 2.4 (Self-Similarity)
A set $K \subset \mathbb{R}^n$ is self-similar (with respect to S) if K is invariant with respect to a finite set of contraction maps S (see Definition A.13) and [Hutchinson, 1981, Sec. 5]

$$H^d(K) > 0, \quad H^d(K_i \cap K_j) = 0, \text{ for } i \neq j, \tag{2.5}$$

where $K = \bigcup_{i=1}^{N} K_i$, $d = \dim_H(K)$ (with $\dim_H(\cdot)$ denoting the Hausdorff-Besicovitch dimension Definition 2.2) and $H^d(\cdot)$ denotes the d-dimensional Hausdorff measure (Definition 2.1). ▲

In definition 2.4, the condition that K is invariant with respect to S ensures that K can be constructed as a finite union of contractions of K. The second condition (2.5) can be seen as a "minimal overlap" condition.

The property of self-similarity (Definition 2.4) is often too idealized to be applicable to fractals found in nature. Rather than the strict self-similarity introduced earlier, these fractals are more likely to exhibit a statistical form of self-similarity.

For a stochastic process, either continuous in time or discrete in time, self-similarity is defined as follows:

Definition 2.5 (Self-Similar Process)
A continuous-time stochastic process $X(t)$ is said to be *self-similar* if the process itself is distributionally indistinguishable from its scaled version $\tau^H X(\frac{t}{\tau})$, where H is the *Hurst exponent* or *self-similarity parameter*, and $\tau > 0$. In other words, $X(t)$ is self-similar if

$$X(t) \sim \tau^H X\left(\frac{t}{\tau}\right).$$

The discrete-time analogue case of this condition for a stochastic process $(X_n)_{n\in\mathbb{Z}}$ can be stated as

$$X_{\tau n} \sim \tau^H X_n,$$

with $\tau \in \mathbb{N}$. ▲

Note that a process $X(t)$ being statistically self-similar implies that both the mean function $\mu_X(t)$ and autocovariance function $\rho_X(t_1, t_2)$ are homogeneous of degree H and $2H$, respectively, since

$$\mu_X(\tau t) = \mathbb{E}\left[f(\tau t)\right] = \mathbb{E}\left[\tau^H f(t)\right] = \tau^H \mu_X(t)$$

and

$$\rho_X(\tau t_1, \tau t_2) = \mathbb{E}\left[f(\tau t_1) f(\tau t_2)\right] = \mathbb{E}\left[\tau^H f(t_1) \tau^H f(t_2)\right] = \tau^{2H} \rho_X(t_1, t_2).$$

Having introduced the fundamentals of fractals and defined the Hurst exponent H in definition 2.5, the manner in which EEG microstate sequences exhibit fractal properties can now be described. The second half of this chapter is structured as follows: Section 2.3 introduces a method for embedding EEG microstate sequences into random walks in order to allow for a Hurst-based analysis of self-similarity. To this end, random walks will be introduced along with their statistical and fractal properties. Section 2.4 introduces a wavelet-based method for estimating the Hurst exponent of a random walk-embedded microstate sequence.

2.3 Random Walk Embedding of EEG Microstates

The aim of this section is to introduce a means of transforming a microstate sequence - a time series on $\{1,2,\ldots,K\}$ - into a time series on $\mathbb{Z}$ in order to assess its fractal properties. A method used in existing studies [Van De Ville et al., 2010] does this by embedding microstate sequences into random walks. The method by which to do so will be the main topic of this section.

In a broad sense, *random walks* refer to a family of random processes that consists of a succession of random steps on some space, e.g. $\mathbb{Z}$ or $\mathbb{R}^2$. For the purposes of this thesis, however, the definition of random walks is restricted to one that is more narrow, in which the random walk is a discrete-time sequence consisting of integer-valued elements, starting with a value of 0 at the origin and taking unit-length steps at each time index. Specifically, the following definition is used:

Definition 2.6 (Random Walk)
Let $U_k \in \{-1,1\}, k = 1,2,\ldots$ be a sequence of random variables referred to as the *steps* or *step sequence*. A *Random Walk* is defined as the process

$$X_n = \sum_{k=1}^{n} U_k, \quad n = 1,2,\ldots,$$

with $X_0 = 0$. ▲

The steps $U_1, U_2, \ldots$ are commonly assumed either iid. or stationary. In this thesis, however, the distribution of the step sequence is deliberately left unaccounted for in Definition 2.6. While making assumptions on $U_1, U_2, \ldots$ would yield a more manageable model, leaving the distribution of the steps unknown allows for the microstate sequence to be "embedded" in the random walk by letting $U_1, U_2, \ldots$ carry information about the microstate sequence. Following the method used by Van De Ville et al. [2010], the microstate classes are first partitioned into two sets. For example, with $K = 4$ microstate classes, a possible partition is $S_1 = \{1,2\}$ and $S_2 = \{3,4\}$. Next, given an estimated sequence of microstates $c_n \in \{1,2,3,4\}, n = 1,2,\ldots,$ let $U_n = 1$ if $c_n \in S_1$ and $U_n = -1$ if $c_n \in S_2$ and compute the random walk as it has been defined in Definition 2.6. The resulting random walk increases by one at every value of n where $c_n \in S_1$ and decreases by one at every value of n where $c_n \in S_2$. The method will be referred to as the *random walk embedding of microstate sequences* (abbr. RW-MS) in this thesis. It is summarized in the following definition:

Definition 2.7 (Random Walk Embedding of EEG Microstates)
Let $(C_n)_{n\in 1,\ldots,N-1}$ be an EEG microstate sequence whose codomain $S \subset \mathbb{N}$ is bounded and of size K. Let S_1 and S_2 be partitions of S satisfying $S_1 \oplus S_2 = S$. The random walk embedding of the microstate sequence (RW-MS) is a discrete process $(X_n)_{n\in 0,\ldots,N-1}$ with $X_0 = 0$ and, for $n = 1,2,\ldots,N-1$,

$$X_n = \sum_{k=1}^{n} U_k, \quad U_k = \begin{cases} 1, & C_k \in S_1, \\ -1, & C_k \in S_2. \end{cases} \tag{2.6}$$

▲

While the RW-MS allows for microstate sequences to be subjected to a Hurst exponent-based analysis of statistical similarity (see Definition 2.5), the question of how the microstate classes should be grouped is one without any apparent answer. Thus, the method involves some arbitrary choice, unless all possible assignments are analyzed in parallel, in which case the data size would increase drastically for larger numbers of microstate classes [von Wegner et al., 2017]. Nonetheless, this method provides a framework for mapping microstate sequences to a metric space, where the concept of statistical self-similarity, as defined in Definition 2.5, can be applied. Understanding some basic properties of random walks will make it more clear how the RW-MS behaves. It will also prove useful later, in Section 2.4, where a wavelet-based method for Hurst exponent estimation will be introduced. Figure 2.4 shows an example of a RW-MS, where a section of the random walk has been scaled to illustrate its similarity to the whole signal.

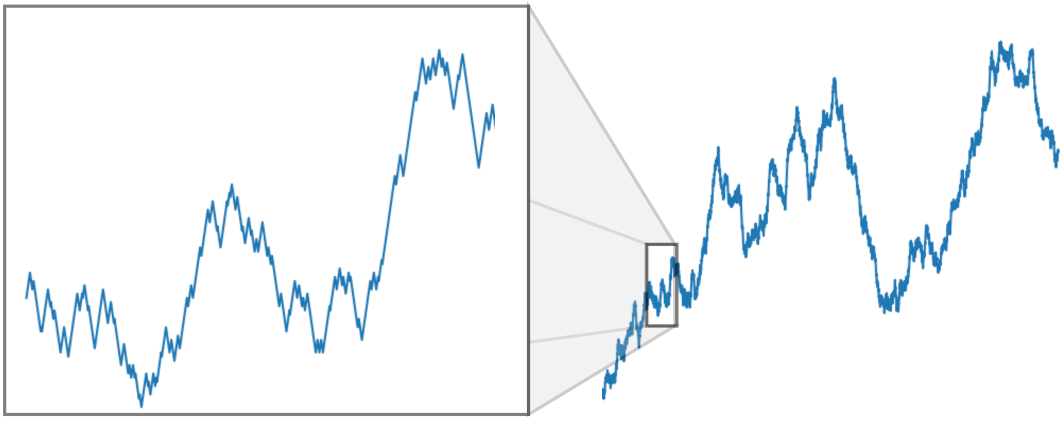


Figure 2.4. Example of a RW-MS along with a scaled version of one of its segment. The resemblance of the two are intended to illustrate statistical self-similarity.

2.3.1 Properties of Random Walks

In the previous section, it was argued that the distribution of the step sequence should be left ambiguous to account for the characteristics of microstate sequences being unknown. This makes it difficult to introduce properties of random walks in an in-depth manner. In order to have a point of comparison, this subsection aims to first introduce two commonly used models for the step sequence before discussing some general properties relating to discrete stochastic processes that will become relevant later, when the Hurst exponent estimation is discussed.

Firstly, one can assume each of the steps $U_1, U_2, \dots$ iid., in which case any given U_k has a probability mass function given as

$$P(U_k = i) = \begin{cases} p, & i = 1 \\ 1 - p, & i = -1 \end{cases}, \quad i \in \{-1, 1\}, k = 1, 2, \dots,$$

with $p \in [0, 1]$. The mean of the step sequence is given as $\mu = \mathbb{E}[U_k] = p - (1 - p)$ for $k = 1, 2, \dots$.

The resulting random walk has first and second moments

$$\begin{aligned}
\mathbb{E}[X_n] &= \sum_{k=1}^{n} \mu = n\mu, \quad n = 1,2,\ldots, \\
\operatorname{Var}[X_n] &= \sum_{i=k}^{n} \operatorname{Var}[U_k] = n\operatorname{Var}[U_1] \\
&= n\left(\mathbb{E}\left[U_1^2\right] - \mathbb{E}[U_1]^2\right) = n(1-\mu^2), \quad n = 1,2,\ldots.
\end{aligned}$$

Noting that $\operatorname{Var}[X_{\tau n}] = \tau n(1-\mu^2) = \tau\operatorname{Var}[X_{\tau n}]$, it appears that, when $\mu = 0$, this random walk is self-similar with Hurst exponent 1/2. Of course, $\tau^{1/2}X_n$ and $X_{\tau n}$ are not truly identical in distribution, as can easily be seen from the fact that the state space of $X_{\tau n}$ is of size $\tau n + 1$, while the state space of $\tau^{1/2}X_n$ is of size $n+1$ and contains integers scaled by $\tau^{1/2}$. This is an inevitable consequence of using fixed step sizes. Since the distance between possible states of τX_n is $\tau^{1/2}$ for any n, this issue becomes less apparent at large n, where a distance of $\tau^{1/2}$ can be considered negligible.

To obtain a slightly more sophisticated model for the step sequence, $U_1, U_2, \ldots$ could be assumed a Markov Chain (Definition A.28), as depicted in Figure 2.5. This implies that the steps U_k have conditional probabilities given as

$$P(U_k = j | U_{k-1} = i) = p_{i,j}, \quad 0 < p_{i,j} = 1 - p_{i,-j} < 1, \quad i,j = \pm 1,$$

for $k = 2,3,\ldots$.

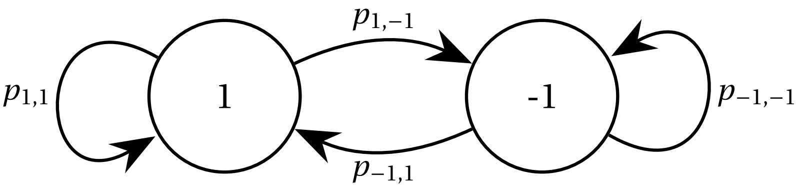


Figure 2.5. Example of steps of a Random Walk modeled as a Markov chain. The numbers indicate transition probabilities.

This Markov chain is irreducible (Definition A.31) and positive recurrent (Definition A.32). Thus, it has a stationary distribution $\pi_j = \lim_{k\to\infty} P(U_{n+k} = j | U_n = i)$ for $i,j = \pm 1$ (Theorem A.34). Assume that the first step of the Markov chain has the probability distribution $P(U_1 = i) = \pi_i$ for $i = \pm 1$. Then, by the law of total probability,

$$P(U_2 = j) = \sum_{i=\pm 1} P(U_2 = j | U_1 = i) P(U_1 = i) = \sum_{i=\pm 1} p_{i,j}\pi_i,$$

which, by definition of the stationary distribution (Definition A.30), is equal to π_j. This indicates that the Markov chain has constant mean $\mu = \pi_1 - \pi_{-1}$. Letting $p_{i,j}^{(k)}$ = denote the k-step transition probability, the autocovariance of the Markov chain is

$$\begin{aligned}
\operatorname{Cov}[U_k, U_{k+m}] = \mathbb{E}[U_k U_{k+m}] - \mu^2 &= \sum_{i\pm 1}\sum_{j\pm 1} j p_{i,j}^{(m)} i\pi_i - \mu^2 \\
&= \sum_{i=\pm 1} \pi_i \left(p_{i,i}^{(m)} - p_{i,-i}^{(m)}\right) - \mu^2, \qquad (2.7)
\end{aligned}$$

which is independent of k. Thus, the Markov chain is stationary (Definition A.26). Note also that since $p_{i,j}^{(m)} \to \pi_j$ for $m \to \infty$, we have $\sum_{i=\pm 1} \pi_i \left(p_{i,i}^{(m)} - p_{i,-i}^{(m)} \right) \to (\pi_1 - \pi_{-1})^2 = \mu^2$, meaning the covariance approaches zero.

The mean of a random walk with the Markov chain step sequence is

$$\mathbb{E}\left[X_n\right] = \sum_{k=1}^{n} \mu = n\mu, \quad n = 1, 2, \dots,$$

The variance is slightly more involved, so outlining the general form will suffice. Let $\rho_i = \mathrm{Cov}\left[U_0, U_i\right]$, given as in (2.7). The variance is given as

$$\begin{aligned}
\mathrm{Var}\left[X_n\right] &= \mathrm{Var}\left[X_{n-1}\right] + \mathrm{Var}\left[U_n\right] + 2\sum_{i=1}^{n-1} \rho_i \\
&= \mathrm{Var}\left[X_{n-2}\right] + \mathrm{Var}\left[U_n\right] + \mathrm{Var}\left[U_{n-1}\right] + 2\left(\sum_{i=1}^{n-1} \rho_i + \sum_{j=1}^{n-2} \rho_j \right) \\
&= \sum_{k=1}^{n} \mathrm{Var}\left[U_k\right] + 2\sum_{i=1}^{n-1} (n-i)\rho_i, \\
&= n(1-\mu^2) + 2\sum_{i=1}^{n-1} (n-i)\rho_i.
\end{aligned}$$

Regardless of the model used for the step sequence, random walks are non-stationary (Definition A.26). Not only does this make intuitive sense, since the state space increases over time, but it can also be easily verified; the mean of a random walk at time $n \in \mathbb{N}$ is $\sum_{k=0}^{n} \mathbb{E}\left[U_k\right]$, which is only constant over all n if the step sequence U_k is a zero-mean process. Given that the step sequence has zero-mean, the variance of X_n is

$$\begin{aligned}
\mathrm{Var}[X_n] &= \mathbb{E}\left[(X_{n-1} + U_n)^2\right], \\
&= \mathrm{Var}[X_{n-1}] + \mathrm{Var}[U_n] + 2\sum_{i=1}^{n-1} \mathrm{Cov}[U_n, U_i], \quad n = 1, 2, \dots,
\end{aligned}$$

which suggests that, in order for the variance to be time invariant, $\mathrm{Var}[U_k] + 2\sum_{i=1}^{k-1} \mathrm{Cov}[U_k, U_i] = 0$ for any k. This is clearly not possible, since if $\mathbb{E}\left[U_1\right] = 0$ and $\mathrm{Var}[U_1] = 0$, we would have $\mathrm{Cov}[U_2, U_1] = 0$ and thus $\mathrm{Var}[U_2] = 0$, implying that $Var\left[U_k\right] = 0$ for all k. Hence the random walk is non-stationary, regardless of the manner in which we define the step sequence U_k.

Consider the *incremental process* $\Delta_k X_n$ given as

$$\Delta_k X_n = X_n - X_{n-k}, \quad k \in \mathbb{N},$$

with $k \le n$. While random walks are non-stationary, the incremental process $\Delta_k X_k$ can be stationary.

Definition 2.8 (Stationary Increments)

A discrete stochastic process $X_0, X_1, \dots$ is said to possess *stationary increments* if its incremental process $\Delta_k X_n$ is stationary (Definition A.26), i.e. if the distribution of $\Delta_k X_n = X_n - X_{n-k}$ is independent of n for any positive integer $k \le n$. ▲

In order to ensure that a random walk has stationary increments, some requirements can be imposed on the step sequence. The mean of the incremental process is

$$\mathbb{E}[\Delta_k X_n] = \mathbb{E}\left[\sum_{i=1}^{n} U_i - \sum_{j=1}^{n-k} U_i\right] = \sum_{i=n-k+1}^{n} \mathbb{E}[U_i].$$

If X_n has stationary increments, this expression must be constant over n for any $0 < k \leq n$. This implies that the step sequence $U_1, U_2, \ldots$ must have constant mean. The covariance of the incremental sequence is

$$\begin{aligned}\mathrm{Cov}[\Delta_k X_n, \Delta_k X_{n+m}] &= \sum_{i=n-k+1}^{n} \sum_{j=n+m-k+1}^{n+m} \mathrm{Cov}\left[U_i, U_j\right], \\ &= \sum_{i=1}^{k} \sum_{j=1}^{k} \mathrm{Cov}\left[U_{i+n-k}, U_{j+n+m-k}\right].\end{aligned}$$

For this to be independent of n, the covariance terms must all be independent of n, i.e. $\mathrm{Cov}[U_1, U_{1+m}] = \mathrm{Cov}[U_n, U_{n+m}]$ for all n. In other words, a random walk $X_0, X_1, \ldots$ has stationary increments if its step sequence is stationary.

Another property that random walks may exhibit is stability, which is defined as follows [Pipiras and Taqqu, 2017].

Definition 2.9 (Stable Process)
A discrete stochastic process is said to be *stable* if it is identical in distribution to linear combinations of independent realizations of the process, up to location and scale parameters. In other words, let $X_{1;1}, X_{1;2}, \ldots$ and $X_{2;1}, X_{2;2}, \ldots$ be two independent copies of the stochastic process $X_1, X_2, \ldots$. The process is said to be stable if, for any $n \in \mathbb{N}$,

$$X_{1;n} + X_{2;n} \sim c_n X_n + d_n, \tag{2.8}$$

with $\sim$ denoting equality in distribution, $c_n > 0$ and $d_n \in \mathbb{R}$. If $d_n = 0$, the process is said to be strictly stable. For symmetric α-stable processes (see Definition A.27), the stability condition (2.8) takes the form[Pipiras and Taqqu, 2017, Eq. (1.5)]

$$X_{1;n} + X_{2;n} \sim 2^{\frac{1}{\alpha}} X_n, \quad \forall n \in \mathbb{Z}, \qquad \blacktriangle$$

The inclusion of α-stable processes in the definition above may seem to be of little relevance at the moment. The wavelet-based method used for Hurst exponent estimation, however, is designed specifically for self-similar symmetric α-stable processes.

In this section, the RW-MS method was introduced as a means of obtaining self-similar sequences from microstate sequences. Random walks and a number of their statistical properties were subsequently discussed. It is now time to introduce the main method for Hurst exponent estimation used in this thesis.

2.4 Wavelet-Based Estimation of the Hurst Exponent

The Hurst exponent was briefly introduced in Definition 2.5, albeit not in an in-depth manner. The aim of this subsection is to introduce a method by Abry et al. [1999] used for determining the Hurst exponent using the wavelet transform and discuss how it may be used to assess the fractal properties of a non-stationary time series. While it may seem as if substantial amounts of wavelet fundamentals are glossed over in this section, relevant theory about the wavelet transform has been relegated to Appendix B to avoid getting sidetracked from the main topic of assessing the fractal properties of microstates.

A process that is statistically self-similar exhibits the scaling formula from Definition 2.5. It follows that since a self-similar process $X(t)$ is identical in distribution to the scaled process $\tau^H X\left(\frac{t}{\tau}\right)$, the wavelet transform (see Definition B.1) of $X(t)$ is identical in distribution to that of $\tau^H X\left(\frac{t}{\tau}\right)$. From (B.1), we have [Simonsen et al., 1997, Eq. (6)]

$$\begin{aligned}
\tilde{X}(a,b) &= \frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty}\psi^*\left(\frac{t-b}{a}\right)X(t)dt \\
&\sim \frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty}\psi^*\left(\frac{t-b}{a}\right)\tau^H X\left(\frac{t}{\tau}\right)dt \\
&\overset{(a)}{=} \tau^H \frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty}\tau\psi^*\left(\frac{t'\tau-b}{a}\right)X\left(t'\right)dt' \\
&= \tau^{H+\frac{1}{2}}\frac{1}{\sqrt{|a/\tau|}}\int_{-\infty}^{\infty}\psi^*\left(\frac{t'-b/\tau}{a/\tau}\right)X\left(t'\right)dt' \\
&= \tau^{H+\frac{1}{2}}\tilde{X}\left(\frac{a}{\tau},\frac{b}{\tau}\right),
\end{aligned} \tag{2.9}$$

where (a) is a result of integration by substitution with $t' = \frac{t}{\tau}$. (2.9) indicates that

$$\tilde{X}\left(\frac{a}{\tau},\frac{b}{\tau}\right) \sim \tau^{-\left(H+\frac{1}{2}\right)}\tilde{X}(a,b). \tag{2.10}$$

This means that for a self-similar process $X(t)$ with Hurst exponent H, its wavelet transform is also self-similar with Hurst exponent $H+\frac{1}{2}$. Note that the $\frac{1}{2}$-term is due to the choice of normalization, and that other choices of normalization could result in a different Hurst exponent of the transformed process.

For the discrete wavelet transform (See Definition B.3), a similar result can be obtained. When applying the discrete wavelet transform on discrete-time signals of finite length, the wavelet coefficients are indexed by two sequences of non-negative integers, called the *octave* and *translation*, respectively. Using 2 as the base for the scale parameter and a translation step size of 1, the discrete wavelet transform of a function $X(t)$ can be stated as

$$\tilde{X}_{j,k} = \frac{1}{\sqrt{2}^j}\int_{-\infty}^{\infty}\psi^*\left(\frac{t}{2^j}-k\right)X(t)dt, \quad j=0,1,\ldots, \quad k\in\mathbb{Z}. \tag{2.11}$$

Letting $\tau = 2^j$, we see that

$$\begin{aligned}
\tilde{X}_{j,k} &= \frac{1}{\sqrt{2}^j} \int_{-\infty}^{\infty} \psi^* \left(\frac{t}{2^j} - k \right) X(t)\, dt \\
&\sim \frac{1}{\sqrt{2}^j} \int_{-\infty}^{\infty} \psi^* \left(\frac{t}{2^j} - k \right) 2^{jH} X \left(\frac{t}{2^j} \right) dt \\
&= 2^{jH} \frac{1}{\sqrt{2}^j} \int_{-\infty}^{\infty} 2^j \psi^* \left(t' - k \right) X(t')\, dt' \\
&= 2^{j(H+\frac{1}{2})} \tilde{X}_{0,k}, \qquad (2.12)
\end{aligned}$$

where we again make use of integration by substitution with $t' = \frac{t}{2^j}$. From (2.12), we see that [Abry et al., 1999, Eq. (4)]

$$\mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right] = j\left(H + \frac{1}{2}\right) + \mathbb{E}\left[\log_2\left(|\tilde{X}_{0,k}|\right)\right],$$

which indicates that an estimate of H can be obtained as the slope of a linear fit of $\mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right]$ versus j, provided that $\mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right]$ is either known or can be estimated. Since the variances of $\log_2\left(|\tilde{X}_{j,k}|\right)$ varies across different octaves j, the weighted least squares is likely to provide better results than the ordinary least squares method. The issue of how the weighted least squares should be applied will be addressed later in this section. First, the issue of obtaining an estimate of $\mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right]$ will be introduced. Although the property of self-similarity in multifractal time series is not central to this thesis, a method that has proven effective in multifractal analysis will be introduced later in Section 2.4.2, such that the monofractality of the random-walk embedded microstate sequences can be assessed.

2.4.1 Averaging Models for Wavelet-Based Hurst Exponent Estimation

Obtaining an accurate estimate of $\mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right]$ is straightforward for α-stable processes with stationary increments. If the process $X(t)$ has stationary increments (Definition 2.8), then, at any octave j, the wavelet coefficients $\tilde{X}_{j,0}, \tilde{X}_{j,1}, \ldots$ form a stationary sequence [Abry et al., 1999]. In this case, a simple estimate of $\mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right]$ can be obtained at any octave j by performing an average along the translation index k. Since the number of wavelet coefficients is limited in practice, we let N_j denote the number of available coefficients at scale j and define [Abry et al., 1999]

$$Y_j = \frac{1}{N_j} \sum_{k=0}^{N_j - 1} \log_2\left(|\tilde{X}_{j,k}|\right), \qquad (2.13)$$

which has the expectation

$$\begin{aligned}
\mathbb{E}\left[Y_j\right] &= j\left(H + \frac{1}{2}\right) + \frac{1}{N_j} \sum_{k=0}^{N_j - 1} \mathbb{E}\left[\log_2\left(|\tilde{X}_{0,k}|\right)\right] \\
&= j\left(H + \frac{1}{2}\right) + \mathbb{E}\left[\log_2\left(|\tilde{X}_{0,0}|\right)\right] \\
&= \mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right], \quad k = 0, 1, \ldots, N_j - 1,
\end{aligned}$$

which suggests that this average is an unbiased estimator for $\mathbb{E}\left[\log_2\left(|\tilde{X}_{j,k}|\right)\right]$ when $X(t)$ has stationary increments.

When $X(t)$ is also a symmetric α-stable process (Definition 2.9), $\log_2\left(|\tilde{X}_{j,k}|\right)$ has finite second-order statistics [Abry et al., 1999]. The estimator has been found to be applicable to any α-stable self-similar process for wavelets with finite order statistics.

Another example of a wavelet-based Hurst estimator is that of Veitch and Abry [1999], which is often used in wavelet-based analysis of processes with long-range dependence. It is based on the observation that (2.12) implies

$$\mathbb{E}\left[|\tilde{X}_{j,k}|^2\right] = 2^{2j(H+\frac{1}{2})}\mathbb{E}\left[|\tilde{X}_{0,k}|^2\right], \quad j \in \mathbb{N}, \text{ and } k = 1,2,\ldots,N_j.$$

In view of this, one could choose to estimate $\mathbb{E}\left[|\tilde{X}_{j,k}|^2\right]$ by averaging across k.

$$Z_j = \frac{1}{N_j}\sum_{k=0}^{N_j-1}|\tilde{X}_{j,k}|^2,$$

for which the expected value is

$$\mathbb{E}\left[Z_j\right] = \frac{1}{N_j}\sum_{k=0}^{N_j-1}\mathbb{E}\left[|\tilde{X}_{j,k}|^2\right] = 2^{j(2H+1)}\mathbb{E}\left[|\tilde{X}_{0,k}|^2\right]. \tag{2.14}$$

This provides a log-linear relationship between Z_j and the octave j, from which H can be estimated.

While (2.14) might seem to indicate that H can be estimated from the slope of a linear fit of $\log_2(Z_{q;j})$ against j, there is one notable issue in taking this approach to the estimation, namely that

$$\mathbb{E}\left[\log_2(Z_j)\right] \neq \log_2\left(\mathbb{E}\left[Z_j\right]\right),$$

which means that the estimate of H obtained from observations of $\log_2(Z_{q;j})$ is biased [Wu, 2020]. In fact, since $\log_2$ is a concave function, Jensen's inequality (See Theorem A.25) implies that $\mathbb{E}\left[\log_2(Z_j)\right] \leq \log_2\left(\mathbb{E}\left[Z_j\right]\right)$, meaning that the estimator tends to underestimate H. To adjust for this bias, one can define [Veitch and Abry, 1999]

$$V_j = \log_2\left(Z_j\right) - g(j),$$

where $g(j)$ is defined such that $\mathbb{E}\left[V_j\right] = \log_2(\mathbb{E}\left[Z_j\right])$. Veitch and Abry [1999] found that the following function $g(j)$ ensures that $\mathbb{E}\left[V_j\right] = \log_2\left(\mathbb{E}\left[Z_j\right]\right)$ [Veitch and Abry, 1999, eq. (17)]:

$$g(j) = \frac{\Gamma'\left(\frac{N_j}{2}\right)}{\Gamma'\left(\frac{N_j}{2}\right)\log(2)} - \log_2\left(\frac{N_j}{2}\right), \tag{2.15}$$

where $\Gamma'(z)/\Gamma(z)$ is the digamma function, i.e. $\Gamma(z)$ is the gamma function and $\Gamma'(z)$ its derivate.

To summarize, (2.13) and (2.14) yields two different models that average the wavelet transform of a stochastic process across different translations at fixed octaves. The expected value of both models provide valuable insight into the Hurst exponent H, provided that the stochastic process has stationary increments. The models and their expected values are given as

$$Y_j = \frac{1}{N_j} \sum_{k=0}^{N_j-1} \log_2\left(|\tilde{X}_{j,k}|\right), \quad \text{with} \quad \mathbb{E}\left[Y_j\right] = j\left(H+\frac{1}{2}\right) + \mathbb{E}\left[\log_2\left(|\tilde{X}_{0,0}|\right)\right], \tag{2.16}$$

$$\begin{aligned} V_j = \log_2\left(\frac{1}{N_j}\sum_{k=0}^{N_j-1}|\tilde{X}_{j,k}|^2\right) - g(j), \quad \text{where} \quad g(j) = \frac{\Gamma'\left(\frac{N_j}{2}\right)}{\Gamma'\left(\frac{N_j}{2}\right)\log(2)} - \log_2\left(\frac{N_j}{2}\right), \\ \text{and} \quad \mathbb{E}\left[V_j\right] = j\,(2H+1) + \log_2\left(\mathbb{E}\left[|\tilde{X}_{0,0}|^2\right]\right). \end{aligned} \tag{2.17}$$

The two approaches are unnamed in the litterature. To make it easier to distinguish between them, Y_j will be called the *average log* model and V_j will be called the *average square* model throughout this thesis.

Both the average log model, Y_j, and the logarithm of the average square model, $\log(Z_j)$, can be computed and plotted against j to obtain an estimate of H. Due to the different variances at different octaves j, however, the weighted least squares method should be chosen over the ordinary least squares method. Since the weights of the weighted least squares should be chosen as the reciprocal of the variances at each octave j, the variances of Y_j and Z_j must first be obtained.

Under the hypothesis that the wavelet coefficients are uncorrelated with each other, Abry et al. [1999] provided a closed-form relationship for the variance of Y_j:

$$\begin{aligned} \mathrm{Var}\left[Y_j\right] &= \mathrm{Var}\left[\frac{1}{N_j}\sum_{k=0}^{N_j-1}\log_2\left(|\tilde{X}_{j,k}|\right)\right] \\ &\overset{(a)}{=} \frac{1}{N_j^2}\mathrm{Var}\left[\sum_{k=0}^{N_j-1}\log_2\left(|2^{j(H+\frac{1}{2})}\tilde{X}_{0,k}|\right)\right] \\ &\overset{(b)}{=} \frac{1}{N_j}\mathrm{Var}\left[\log_2\left(|\tilde{X}_{0,0}|\right)\right], \end{aligned} \tag{2.18}$$

where (a) follows from using the distributional identity (2.12), and (b) is true if the original process $X(t)$ has stationary increments and the wavelet coefficients are uncorrelated across k. $\mathrm{Var}\left[\log_2\left(|\tilde{X}_{0,0}|\right)\right]$ can be estimated by the sample variance. Note that since $\mathrm{Var}\left[\log_2\left(|\tilde{X}_{0,0}|\right)\right] = \mathrm{Var}\left[\log_2\left(|\tilde{X}_{j,0}|\right)\right]$ for any j, one may also sample the variance at any other scale.

The variance of the average square model V_j is identical to that of $\log_2(Z_j)$, since the variance is unaffected by the addition of a deterministic term $g(j)$. However, due to $\log_2(Z_j)$ having the order of logarithm and summation reversed relative to Y_j, obtaining the variance of $\log_2(Z_j)$ is more complicated. Under the assumption that the wavelet coefficients are uncorrelated across both time and scale, Veitch and Abry [1999] found that

$$\mathrm{Var}\left[V_j\right] = \mathrm{Var}\left[\log_2\left(Z_j\right)\right] = \frac{\zeta\left(2, \frac{N_j}{2}\right)}{\log^2(2)}, \tag{2.19}$$

where $\zeta(\cdot)$ is the generalized Riemann Zeta function $\zeta(z,n) = \sum_{k=0}^{\infty}(n+k)^{-z}$.

In practice, the wavelet coefficients can rarely be considered truly uncorrelated; instead, the covariances between different wavelet coefficients are assumed negligible when they vanish sufficiently rapidly. As a result, the performance of the estimator is negatively affected by the long-range dependence exhibited by self-similar processes with Hurst exponents $H > \frac{1}{2}$. A value of H close to 1 suggests a significant level of long-range dependence of the signal, which, at any fixed octave j, could introduce correlation between the $\log_2 \left(\tilde{X}_{j,k}|\right), \quad k = 0, 1, \ldots, N_j - 1$.

In order to reduce the effect of long-range dependence on the wavelet coefficients and strengthen the validity of the decorrelation hypothesis used in (2.18), one can choose to use a mother wavelet for which the number of *vanishing moments* (See Definition B.6) is large [Abry et al., 1999], i.e. wavelets $\phi(\cdot)$ for which

$$\int_{-\infty}^{\infty} t^j \phi(t) dt = 0, \quad j = 0, 1, \ldots, N-1,$$

in which case we say that the wavelet $\phi(\cdot)$ has $N \in \mathbb{N}$ vanishing moments if N is the largest value for which the equation holds. The Daubechies wavelets, which can be implemented as a series of filters with a finite number of taps, achieve the maximal amount of vanishing moments for a given support width - with $2N$ taps N vanishing moments. This makes them ideal for wavelet-based Hurst estimation.

Having obtained a closed form expression of the variances of both Y_j and V_j, the Hurst exponent can now be estimated. The weighted least squares Hurst exponent estimators $\hat{H}_Y$ and $\hat{H}_V$ - based on the average log model Y_j and the average square model V_j, respectively - are defined over a range of octaves $j_0 \le j \le j_1$ as [Abry et al., 1999; Veitch and Abry, 1999]

$$\hat{H}_Y = \sum_{j=j_0}^{j_1} w_j Y_j - \frac{1}{2}, \tag{2.20}$$

$$\hat{H}_V = \frac{\sum_{j=j_0}^{j_1} w_j V_j - 1}{2}, \tag{2.21}$$

where, in either model, the weights w_j take the general form

$$w_j = \frac{S_0 j - S_1}{\mathrm{Var}\left[X_j\right] (S_0 S_2 - S_1{}^2)} \quad \text{with} \quad S_m = \sum_{i=j_1}^{j_2} \frac{i^m}{\mathrm{Var}\left[X_i\right]}, \quad m = 0, 1, 2,$$

where X_j is either Y_j or W_j, depending on which estimator is chosen. The performance of the two models will be assessed on a simple example later, in Section 3.1.

2.4.2 Wavelet-Based Multifractal Analysis

This thesis is primarily concerned with monofractal time series, for which the Hurst exponent is a constant. For monofractal time series, the power-law $X(\tau t) = \tau^H X(t)$, imposed by the self-similarity property, is independent of There are cases in which the self-similar scaling of a fractal

Testing the time series for multifractal behavior would provide an argument for the validity of the monofractal assumption - or lack thereof. If the process does exhibit multifractality, the monofractal assumption should be reevaluated; if it does not, the Hurst exponent sufficiently captures the fractal properties of the process.

A model commonly used for assessing multifractal properties is one similar to Z_j from (2.17), in which the exponent can take any non-zero integer power q rather than being limited to 2. In practice, such a model would be unstable for $q < 0$, since any wavelet coefficient could randomly take a value close to zero, and subsequently be raised to a negative power. To mitigate this issue, the model is computed from the *wavelet leaders* of a signal; they are defined as follows [Jaffard, 2004, Defs. 10-11]:

Definition 2.10 (Wavelet Leaders)
Let $X(t) \in L^2(\mathbb{R})$ be a real-valued function with a discrete wavelet transform $\tilde{X}_{j,k}$, whose basis functions have compact support (see Definition A.7) and form an orthonormal basis of $L^2(\mathbb{R})$. Let $\lambda_{j,k} = \left[2^j k, (k+1)2^j\right)$ and, with slight abuse of notation, $3\lambda_{j,k} = \lambda_{j,k-1} \cup \lambda_{j,k} \cup \lambda_{j,k+1}$. The wavelet leader at octave $j = 0, 1, \ldots$ and location $k \in \mathbb{Z}$ is given as [Jaffard, 2004; Wendt et al., 2007]

$$L_{\tilde{X};j,k} = \sup_{\lambda_{j',k'} \subset 3\lambda_{j,k}} |\tilde{X}_{j',k'}|. \qquad \blacktriangle$$

From (2.11) we see that within the subinterval $t \in \lambda_{j,k}$ of the convolution, the mother wavelet $\phi^*(t')$ takes arguments $t' \in [0,1)$ for any $j, k \in \mathbb{Z}$. This suggests that wavelet functions, $\phi_{j,k}(t) = \phi(2^{-j}t - k)$, and their corresponding coefficient, $\tilde{X}_{j,k}$, can be considered 'localized' within the interval $\lambda_{j,k}$ - albeit only in a heuristic sense, since the support of the mother wavelet $\phi^*(t)$ may not be $[0,1)$.

Note that the set of intervals that satisfy $\lambda_{j',k'} \subset 3\lambda_{j,k}$ in Definition 2.10 includes not only $\lambda_{j,k-1}$, $\lambda_{j,k}$ and $\lambda_{j,k+1}$, but also a number of wavelet coefficients at octaves lower than j, since $\lambda_{j,k} = \lambda_{j-1,2k} \cup \lambda_{j-1,2k+1}$. Thus, the wavelet leader $L_{\tilde{X};j,k}$ is the largest absolute wavelet coefficient among all $\tilde{X}_{j',k'}$ localized within $3\lambda_{j,k}$, including those at finer resolutions, where $j' \leq j$ [Wendt et al., 2007].

Letting $N_{L;j}$ denote the number of wavelet leaders available at at octave j, the wavelet leader-based approach considers the *structure function* given as

$$S_{j,q} = \frac{1}{N_{L;j}} \sum_{k=0}^{N_{L;j}} |\tilde{X}_{j,k}|^q, \tag{2.22}$$

with $q, j \in \mathbb{Z}$. Note that if (2.22) had been computed on wavelet coefficients rather than wavelet leaders, any wavelet coefficient close to zero could affect the sum drastically for negative powers q. By using the wavelet leaders rather than the wavelet coefficients in (2.22), the structure function can be computed in a stable manner for negative powers q [Jaffard, 2004, Sec. 4.4].

In the limit $j \to \infty$, the structure function is [Wendt et al., 2007; Jaffard, 2004]

$$S_{j,q} = c_q 2^{j\xi(q)},$$

where $c_q \in \mathbb{R}$ is some scalar and the *scaling function* $\xi(q)$ is concave for multifractal processes [Jaffard et al., 2007] and affine for monofractal processes. Thus, when the characteristics of the time series are largely unknown, an estimation of $\xi(q)$ using log-linear regression provides a means of assessing whether the time series is likely to be monofractal; if the estimate of $\xi(q)$ follows a strong linear trend, the time series is likely monofractal.

Although the microstate sequences and their random-walk embeddings are discrete in time, the process $X(t)$ has been assumed a continuous-time signal in this section. While the discrete wavelet transform maps a continuous function to a discrete set of coefficients using a family of continuous-time basis functions, the findings of Daubechies [1988] provides a discrete scheme of filters with a finite number of taps that is equivalent to a discrete wavelet transform with orthonormal basis functions. This scheme is entirely discrete by design; it can be applied to sampled signals without compromising its theoretical framework. Additionally, the Daubechies wavelets are designed as to have the highest number of vanishing moments for a given number of filter taps, which should mitigate the extent to which long-range dependence affects the performance of the wavelet-based Hurst estimator. The topic of Hurst exponent estimation using Daubechies wavelets will be addressed in the following subsection. An in-depth summary of Daubechies wavelets and multiresolution analysis can be found within Appendix B.

2.4.3 Hurst Exponent Estimation with Daubechies Wavelets

Section 2.4.1 introduced a method for estimating the Hurst exponent of a process from its wavelet transform, provided the process and its wavelet coefficients fulfill a set of conditions. The method leverages the fact that the wavelet coefficients of a self-similar process with stationary increments are identical in distribution across octaves $j \in \mathbb{Z}$ up to a scale factor $2^{j(H+1/2)}$. Thus, from the log-linear behavior exhibited by the means of the wavelet coefficients at different octaves, an estimate of H can be obtained by performing a linear fit on the logarithm of the wavelet mean across a range of octaves. This requires an estimate of the means; two models, (2.16) and (2.17), were introduced for this purpose.

Until this point, any coefficient of the discrete wavelet transform has been introduced without reference to any specific choice of wavelet function; for future sections, however, the chosen wavelet will be of the Daubechies family (see Appendix B.2), first introduced in Daubechies [1988]. This subsection aims to fill in gaps in earlier sections caused by the lack of specificity on the choice of wavelets.

The fundamentals of Daubechies wavelets will first be summarized, before delving into their implementation. Afterwards, the entire process of estimating the Hurst exponent will be summarized in a step-by-step guide to wavelet-based Hurst estimation. This section provides only a simplified overview of Daubechies wavelets; the topic of wavelets is discussed in more detail in Appendix B, where Appendix B.2 concerns the construction of Daubechies wavelets.

2.4.3.1 Summary of Daubechies Wavelets and Multiresolution Analysis

The Daubechies wavelets are based on the multiresolution analysis framework of Mallat [1989]. Multiresolution analysis allows for $L^2(\mathbb{R})$-functions to be written as the limit of successive smoothed versions of the function, where an increasingly concentrated smoothing function gradually introduces more fine-scale features of the original function. While Appendix B.1.1 provides a more detailed description of multiresolution analysis, an abridged version will suffice here.

A multiresolution analysis of the set of square-integrable functions, $L^2(\mathbb{R})$ (Definition A.2), consists of a sequence $V_j, j \in \mathbb{Z}$ of closed subspaces of $L^2(\mathbb{R})$, for which

$$L^2(\mathbb{R}) = \overline{\bigcup_{j\in\mathbb{Z}} V_j}, \tag{2.23}$$

where the overline denotes the closure operator. The 'multiresolution' aspect is introduced by imposing certain conditions on V_j; most notably, that $V_j \subset V_{j-1}$ and $f(t) \in V_j \implies f(2t) \in V_{j-1}$ for all j. These two conditions (heuristically) suggest that the orthogonal projections of an $L_2(\mathbb{R})$-function onto V_j and V_{j-1} yield approximations of the function at different resolutions, with the latter being at a level of detail 'twice' that of the former. Additional conditions are listed in Definition B.4, but skipped here.

Since $V_j \subset V_{j-1}$, we have $V_{j-1} = V_j \bigoplus W_j$, with W_j denoting the orthogonal projection of V_{j-1} onto the complement of V_j. W_j contains the additional information provided by an approximation at the $(j-1)$th octave compared to one at jth octave. The subspaces $W_j, j \in \mathbb{Z}$ allow us to restate (2.23) in terms of the orthogonal subspaces starting from V_i, for any given $i \in \mathbb{Z}$:

$$L^2(\mathbb{R}) = V_i \oplus \overline{\bigoplus_{j=-\infty}^{i} W_j}, \tag{2.24}$$

where the overline denotes the closure operator.

For the asymptotic case of (2.24) when $i \to \infty$ and $V_\infty = \{0\}$, Mallat [1989] proved the existence of a function $\psi(t)$ for which $\left\{2^{-j/2}\phi(2^{-j}t-k)\right\}_{j,k\in\mathbb{Z}}$ is an orthonormal basis of $L(\mathbb{R})$. As will be shown shortly, however, the bounded model (2.24) lends itself well to understanding some of the practical aspects of Daubechies wavelets.

The discrete wavelet transformation by Daubechies [1988] decomposes the signal of interest by iteratively projecting its component from one space, V_j, onto its two orthogonal subspaces, V_{j+1} and W_{j+1}. The method is designed for discrete-time signals, $(x_n)_{n\in\mathbb{Z}} \in \ell^2(\mathbb{Z})$; multiresolution analysis requires these signals to be 'translated' into functions on $\mathbb{R}$. This is achieved by first redefining $(c_n^{(0)})_{n\in\mathbb{Z}} = (x_n)_{n\in\mathbb{Z}}$ and choosing the spaces V_j and functions $\phi(\cdot)$ such that $\left\{2^{-j/2}\phi(2^{-j}t-k)\right\}_{k\in\mathbb{Z}}$ is an orthonormal basis of V_j. Next, consider the function [Daubechies, 1988, Sec. 2.C]

$$f(t) = \sum_{n\in\mathbb{Z}} c_n^{(0)}\phi(t-n), \tag{2.25}$$

which is a linear combination of the basis vectors of V_0, meaning $f(\cdot) \in V_0$. The function $f(\cdot)$ can be decomposed into its components in V_1 and W_1, and its component in V_1 can be further decomposed into its components in V_2 and W_2. After $i > 0$ repeated steps of this procedure, $f(\cdot)$ has been separated into its components in the subspaces listed in (2.24), albeit with non-zero components only at octaves $j < 0$.

At every $j < 0$, sequences $(c_n^{(j)})_{n\in\mathbb{Z}}$ and $(d_n^{(j)})_{n\in\mathbb{Z}}$ can be defined, respectively, from $f(\cdot) \in V_j$ and $f(\cdot) \in W_j$ analogues to (2.25). Expressing the $(c_n^{(j)})_{n\in\mathbb{Z}}$ and $(d_n^{(j)})_{n\in\mathbb{Z}}$ in terms of $(c_n^{(j-1)})_{n\in\mathbb{Z}}$ reveals how the discrete wavelet transform is implemented with Daubechies wavelets: For a given signal $(c_n^{(0)})_{n\in\mathbb{Z}}$ and sequences $(h_n)_{n\in\mathbb{Z}}$ and $(g_n)_{n\in\mathbb{Z}}$ with finite support, the Daubechies wavelet coefficients are obtained as [Daubechies, 1988, Eq. (2.47),(2.48)]

$$\begin{aligned} c_k^{(j)} &= \sum_{n\in\mathbb{Z}} h_{n-2k} c_n^{(j-1)}, \\ d_k^{(j)} &= \sum_{n\in\mathbb{Z}} g_{n-2k} c_n^{(j-1)}, \end{aligned} \tag{2.26}$$

where the previous sequence, $(c_k^{(j-1)})_{k\in\mathbb{Z}}$, can be discarded after any step, j. The filter taps $(h_n)_{n\in\mathbb{Z}}$ and $(g_n)_{n\in\mathbb{Z}}$ are defined according to a set of conditions that impose an underlying wavelet transform with an orthonormal basis of compactly supported wavelets, whose order of regularity increases linearly with the support width. One of these conditions relates $(g_n)_{n\in\mathbb{Z}}$ to $(h_n)_{n\in\mathbb{Z}}$ as $g_n = (-1)^n h(-n+1)$, allowing both functions to be defined by a common set of coefficients. Some of the other conditions are described in Appendix B.2.

The filter taps are computed such that they guarantee the maximal number K of vanishing moments for the given filter length. It has been found that, when the filters $(h_n)_{n\in\mathbb{Z}}$ and $(g_n)_{n\in\mathbb{Z}}$ are both of length $2K$, the wavelet transform has K vanishing moments [Daubechies, 1988]. For the sake of convenience, it is assumed that the filters are supported on $n = 0, 1, \dots, 2N-1$.

2.4.3.2 Practical Considerations

For the purposes of this thesis, Daubechies wavelets with five vanishing moments are used via the PyWavelets package [Lee et al., 2009] for the programming language Python 3.8.

Since the real signals are finite length signals on a bounded set of integers $N = 0, 1, \dots, N-1$, the length N_j of the sequence at the jth step of the decomposition is finite as well. Since the $2k$-terms in (2.26) effectively introduce a downsampling by a factor 2 at each step, it follows that the lengths of the signal at different j should also decrease approximately by a factor 2 at each step. When disregarding edge cases and assuming the original signal is of length $N = 2^i k$ for some $k \in \mathbb{N}$, the Daubechies wavelet transform of the signal, terminated at the ith step, has $\frac{N}{2} + \frac{N}{4} + \cdots + \frac{N}{2^i} + \frac{N}{2^i} = N$ coefficients, where the final term is repeated twice due to i being the only octave j where $(c_n^{(j)})_n$ is not discarded. Thus, the Daubechies wavelet transform of a signal preserves (in theory) the number of entries, as was to be expected from a decomposition onto an orthonormal basis [Daubechies, 1988].

Due to the edge effects caused by applying a filter of length $2K$ to a finite-length signal, the length of either sequence given in (2.26), denoted N_j, is equal to the number of entries for which the filter does not, at any point during the convolution, exceed the boundaries of the previous sequence, $(c_n^{(j-1)})_{n=0,1,\ldots,N_{j-1}}$. In other words, N_j is the total number of integers k that satisfy $[2k, 2(K+k)-1] \subseteq [0, N_{j-1}-1]$; the condition is satisfied by $k = 0, 1, \ldots, \lfloor N_{j-1}/2 - K \rfloor$, which implies $N_j = \lfloor N_{j-1}/2 - K + 1 \rfloor$ (assuming, of course, that N_j is positive).

Neglecting to account for the edge cases of a signal results in information being lost during the decomposition, making exact reconstruction impossible. The aim of this thesis is to assess general characteristics of potentially self-similar signals from their wavelet representations; local features at either end of the signals are negligible, and signal reconstruction quality is a non-issue, since In other words, the amount of lost information is unlikely to cause any major issues.

To account for edge effects, signals are often extended past their end points using padding prior to the decomposition, resulting in the number of wavelet coefficients being equal to or greater than the length of the original signal. This generally introduces some redundancy in the signal, and could - depending on the extension mode chosen - greatly affect how the dynamics of the sequence are observed, especially at later steps, where the number of coefficients is lower. Pywavelets implements the discrete wavelet transform using a cascading filter banks algorithm, which is computationally efficient for inputs whose length is a power of 2. Thus, the wavelet decomposition in PyWavelets requires the use of extension modes.

The extension mode should be chosen as to best emulate the behavior of the signal. For this thesis, a potential candidate is the 'antireflect' mode [Lee et al., 2009]

$$\ldots,\ (2x_0 - x_2),\ (2x_0 - x_1)\ \mid\ x_0,\ x_1,\ \ldots,\ x_{N_j-1}\ \mid\ \left(2x_{N_j-1} - x_{N_j-2}\right),\ \left(2x_{N_j-2} - x_{N_j-3}\right), \ldots. \tag{2.27}$$

At every step, a discrete wavelet transform (with filter length $2K$) using this mode outputs $N_j = \lfloor \frac{1}{2}\left(N_{j-1} + -1\right) \rfloor$ coefficients.

To explain why this mode is an intuitive choice, recall the nature of the random walks from Section 2.3, in which $X_n = X_{n-1} + U_n$ for some random step U_n. Consider the right-side extension; if we assume $(x_n)_{n=0,1,\ldots}$ is an observation of a random walk with steps $(u_n)_{n=0,1,\ldots}$, then $2x_{N_j-1} - x_{N_j-2} = x_{N_j-1} + u_{N_j-1}$. Furthermore, if the steps are iid., then $\mathbb{E}\left[X_{N_j-1} + U_{N_j-1}\right] = \mathbb{E}\left[X_{N_j}\right]$. This suggests that the mode yields a unbiased estimate of X_{N_j}, with respect to the prior probability; when conditioned on an observation, x_{N_j-1}, the unbiased estimate would be $x_{N_j-1} + \mathbb{E}\left[U_{N_j}\right]$. The variance of the padded value do not match that of $X_{N_{j-1}+U_j}$, since $\text{Var}\left[U_{N_j-1}\right] = \text{Cov}\left[X_{N_j-1}, U_{N_j-1}\right] \neq \text{Cov}\left[X_{N_j-1}, U_{N_j}\right] = 0$. The left-side extension imposes antisymmetry about x_0, but it cannot be meaningfully linked to random-walks, since random-walks are left-bounded at zero. Nonetheless, this mode has been chosen for this thesis.

2.4.4 Final Wavelet-Based Model for Hurst Estimation

Having discussed several topics related to the implementation of the wavelet-based Hurst exponent estimator, it is time to conclude this chapter by summarizing the steps involved in obtaining a Hurst estimate using the wavelet transform.

The procedure can be summarized as follows:

1. Let $(x_n)_{n=0,1,\dots,N-1}$ denote the observed self-similar (Definition 2.5) time series.
    a) Assume it is stable (Definition 2.9) and has stationary increments (Definition 2.8).
    b) For a microstate sequence $(c_n)_{n\in 1,\dots,N-1}$ with elements in a set $S \subset \mathbb{N}$ of size K, a random-walk embedding, (2.6), is first applied by computing $x_n = \sum_{k=1}^{n} u_k$, where $u_k = (-1)^{i-1}$ for $c_n \in S_i$, with $S_1 \oplus S_2 = S$ and $x_0 = 0$.
2. The discrete wavelet transform of $(x_n)_{n=0,1,\dots}$ is obtained by iterating

$$c_k^{(j)} = \sum_n h_{n-2k} c_n^{(j-1)}, \qquad d_k^{(j)} = \sum_n g_{n-2k} c_n^{(j-1)},$$

over a range of steps $j = 1, 2, \dots$.

    a) The taps $h_{n-2k} = (1)\, g_{-n-2k}$ are supported on $0, 1, \dots, 2K$, for a filter length $2K$.
    b) The sequences $(c_k^{(j-1)})_{k=0,1,\dots,N_j}$ can be discarded after step j.
    c) Since both outputs from each iteration is of length $N_j \approx N_{j-1}/2$, the process must be terminated while $N_j > 0$.
    d) The use of signal extension modes is required for optimal computational efficiency.
3. From the set of wavelet coefficients at levels $j_1 \le j \le j_2$, $\left\{(d_k^{(j)})_{k=0,1,\dots,N_j}\right\}_{j_1 \le j \le j_2}$, one of the two following models can be computed:

$$Y_j = \frac{1}{N_j} \sum_{k=0}^{N_j-1} \log_2\left(|d_k^{(j)}|\right), \qquad V_j = \log_2\left(\frac{1}{N_j} \sum_{k=0}^{N_j-1} |d_k^{(j)}|^2\right) - \left(\frac{\Gamma'\left(\frac{N_j}{2}\right)}{\Gamma'\left(\frac{N_j}{2}\right)\log(2)} - \log_2\left(\frac{N_j}{2}\right)\right).$$

4. From either Y_j or V_j, an estimate of the Hurst exponent can be obtained as, respectively,

$$\hat{H}_Y = \sum_{j=j_0}^{j_1} w_j Y_j - \frac{1}{2}, \qquad \hat{H}_V = \frac{\sum_{j=j_0}^{j_1} w_j V_j - 1}{2}.$$

    a) For X_j denoting either Y_j or V_j, the weights are $w_j = \left(S_0 j - S_1\right) / \left(\mathrm{Var}\left[X_j\right] (S_0 S_2 - {S_1}^2)\right)$ with $S_m = \sum_{i=j_1}^{j_2} i^m / \mathrm{Var}\left[X_i\right]$.
    b) Computing the weights requires an estimate of the variances, which is obtained for Y_j by using (2.18) (on sample variances of $|d_k^{(j)}|$), and for V_j by using (2.19).

Simulation Studies 3

The previous chapter introduced a Hurst exponent estimation scheme for discrete-time signals based on the discrete wavelet transform, but the performance of the transform was not assessed. Additionally, no other estimation methods were introduced to serve as a point of comparison. This chapter aims to remedy both of these issues, by introducing a deep learning-based approach to Hurst exponent estimation and comparing it to the wavelet-based approach.

First, the performance of the wavelet-based estimators will be applied to a simple family of self-similar processes known as the discrete Wiener process, whose Hurst exponent is $H = 5$. By analyzing the MSE, bias and variance of the estimator over different pairings of observed scale boundaries $j_0 \leq j \leq j_1$ and averaging models Y_j and V_J (introduced in Section 2.4.1), model parameters will be chosen.

Section 3.2 concerns the issue of designing a deep learning-based model using supervised learning. An immediate issue of using supervised learning for Hurst estimation is that no ground truth is available for microstate sequences; the network would require training labels from the wavelet-based method, presumably causing it to replicate the biases and inaccuracies of the wavelet-based method. Additionally, a sizable amount of microstate sequences would be required to train such a network; a much easier approach would be to train a model on simulated data and compare its performance against the wavelet-based model on sets of data, whose Hurst exponents are known. For this purpose, a neural network is instead designed for Hurst exponent estimation of *fractional Brownian motion* - a generalized Wiener process whose Hurst exponent can take any value $H \in [0, 1)$.

The last section, Section 3.3, seek to better understand the underlying dynamics of the EEG microstate model. The aim of Section 3.3 is to obtain a model that can be used to generate synthetic EEG data from a given microstate sequence. This section is driven by inductive reasoning and conjecture, and some known characteristics of EEG microstates are disregarded in the interest of obtaining a model, whose statistics are adjustable or easily determined.

3.1 Hurst Exponent Estimation of a Wiener Process

The aim of this section is to provide an example of Hurst exponent estimation using the wavelet-based method introduced in the previous chapter. From the results of this section, the model is fine-tuned for use in later chapters.

In order to assess the performance of the wavelet-based Hurst estimation scheme, a discrete-time self-similar stochastic process is required. One of the most simple models for such a process is the *discrete Wiener process*, which can be defined as follows [Kay, 2006, Example 20.3]:

Definition 3.1 (Discrete Wiener Process)
The discrete Wiener process $(X_n)_{n\in\mathbb{N}}$ is a sequence of random variables given as

$$X_n = \sum_{k=0}^{n} U_k, \quad n = 0, 1, \dots,$$

where $U_k \overset{iid.}{\sim} \mathcal{N}(0, \sigma^2)$. ▲

Clearly, the discrete Wiener process has zero mean. Its autocovariance is

$$\begin{aligned}\operatorname{Cov}\left[X_{n_1}, X_{n_2}\right] &= \mathbb{E}\left[\sum_{j=0}^{n_1} U_j \sum_{k=0}^{n_2} U_k\right] \\ &= \mathbb{E}\left[\sum_{j=0}^{n_1}\sum_{k=0}^{n_2} U_j U_k\right] \\ &= \sum_{k=0}^{\min(n_1,n_2)} \mathbb{E}\left[U_k^2\right] \\ &= \sigma^2 \min(n_1, n_2).\end{aligned}$$

It is also self-similar, since, for any positive integer τ, we have

$$\operatorname{Cov}\left[X_{\tau n_1}, X_{\tau n_2}\right] = \sigma^2 \tau \min(n_1, n_2) = \tau^{\frac{1}{2}} \operatorname{Cov}\left[X_{n_1}, X_{n_2}\right],$$

indicating that the process has a Hurst exponent $H = 1/2$.

The discrete Wiener process is easily generated from partial sums of N Gaussian variables. While its Hurst exponent being fixed at $1/2$ may affect the comprehensiveness of the test, the model provides a simple method for parameter adjustments.

For the experiment in this section, 2^{12} signals of length 2^{14} were used, and the variance σ^2 of Definition 3.1 was set to 1. An example of one of the generated Wiener processes is shown in Figure 3.1. Daubechies wavelets with 5 vanishing moments were used for this test. Both of the wavelet average models introduced in Section 2.4.1 were used for the wavelet-based method during this test.

The two models were the average-log model, Y_j, whose associated Hurst exponent estimate was named $\hat{H}_Y$, and the average-square model V_j, whose Hurst exponent estimate was named

$\hat{H}_V$. The mean squared error, sample bias and sample variance were computed for both models, using every possible combination of boundaries, j_0 and j_1, for the set of octaves included in the linear regression. The results are given in Table 3.1 and Table 3.2 for $\hat{H}_Y$ and $\hat{H}_V$, respectively. Each row shows the optimal result in terms of MSE, at the given lower bound of observed octaves, j_0.

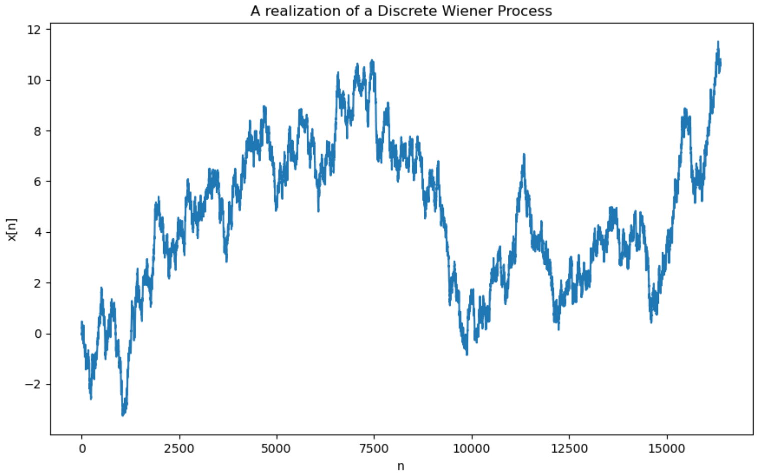


Figure 3.1. A realization of a Wiener process.

Table 3.1. Hurst exponent estimates $\hat{H}_Y$ were computed for 2^{12} signals of length 2^{14} for every possible pair of bounds j_0 and j_1 of the range of octaves $j_0, j_0+1, \ldots, j_1$ over which H is estimated. At each j_0, the j_1 of the estimate with minimal MSE is given along with averages of MSE, bias and variance across the estimates. Absolute minimum within each category is marked.

j_0	j_1	MSE	Bias	Variance
1	10	7.548×10^{-3}	-8.614×10^{-2}	1.273×10^{-4}
2	9	9.467×10^{-4}	-2.638×10^{-2}	2.508×10^{-4}
3	9	7.860×10^{-4}	-1.523×10^{-2}	5.542×10^{-4}
4	9	1.459×10^{-3}	-1.361×10^{-2}	1.273×10^{-3}
5	9	3.423×10^{-3}	-1.035×10^{-2}	3.315×10^{-3}
6	10	8.260×10^{-3}	3.053×10^{-3}	8.251×10^{-3}
7	10	1.880×10^{-2}	2.070×10^{-2}	1.837×10^{-2}
8	10	4.855×10^{-2}	4.476×10^{-2}	4.655×10^{-2}
9	10	1.996×10^{-1}	8.595×10^{-2}	1.922×10^{-1}

Table 3.2. Hurst exponent estimates $\hat{H}_V$ were computed for 2^{12} signals of length 2^14 for every possible pair of bounds j_0 and j_1 of the range of octaves $j_0, j_0+1, \ldots, j_1$ over which H is estimated. At each j_0, the j_1 of the estimate with minimal MSE is given along with averages of MSE, bias and variance across the estimates. Absolute minimum within each category is marked.

j_0	j_1	MSE	Bias	Variance
1	10	6.612×10^{-3}	-8.088×10^{-2}	7.100×10^{-5}
2	10	4.039×10^{-4}	-1.529×10^{-2}	1.700×10^{-4}
3	9	3.120×10^{-4}	-3.661×10^{-3}	2.986×10^{-4}
4	8	6.475×10^{-4}	-1.713×10^{-3}	6.446×10^{-4}
5	8	1.843×10^{-3}	4.204×10^{-3}	1.825×10^{-3}
6	9	5.442×10^{-3}	2.363×10^{-2}	4.884×10^{-3}
7	9	1.686×10^{-2}	4.309×10^{-2}	1.500×10^{-2}
8	10	5.022×10^{-2}	1.145×10^{-1}	3.712×10^{-2}
9	10	1.487×10^{-1}	1.774×10^{-1}	1.173×10^{-1}

The results in Table 3.1 and **??** indicate that including the wavelet coefficients of octaves $j_0 = 3$ through j_9 in the Hurst exponent estimation for signals of length 2^{14} appears to provide the optimal performance for both $\hat{H}_Y$ and $\hat{H}_Y$. The subpar performance at low values of j_0 is likely related to the fact that wavelet coefficients at the lower range of octaves are not as heavily processed, and cover narrow time intervals, making them more vulnerable to noise and errors. At larger octaves, most of the noise present in the original signal has likely been reduced by successive application of the filter used in the Daubechies wavelet decomposition.

These results are mostly in line with what has been found in other studies; Wu [2020] also found that an increase in j_0 yielded a decrease in performance, and argued that the loss of information caused by discretely sampling a continuous function introduces errors that are significant at smaller octaves, but decreases at higher octaves.

Moving forward, $j_0 = 3$ and $j_1 = 9$ are chosen as boundaries for both the average-log model and average-squares model when the data is of length 214. If need be, the lower boundaries will be the first to be adjusted, should the method cause problems in the future.

3.2 Convolutional Neural Network-Based Hurst Exponent Estimation of Fractional Brownian Motion

This section delves into the issue of how well a convolutional neural network (abbr. CNN, see Appendix C) performs compared to the wavelet-based method in estimating the Hurst exponent of a time series. For this purpose, *fractional Brownian motion* provides a means of generating a time series for which the Hurst exponent is easily controlled. The following subsection serves the purpose of introducing the basics of fractional Brownian motion, including how it is simulated.

3.2.1 Fractional Brownian Motion

A *fractional Brownian motion* (abbr. fBm) with Hurst exponent H is a zero-mean Gaussian process $B_H(t)$ for which - when standardized - the autocovariance is on the form [Dieker and Mandjes, 2003]

$$\rho_B(t_1, t_2) = \mathbb{E}\left[B_H(t_1)B_H(t_2)\right] = \frac{1}{2}\left({t_1}^{2H} + {t_2}^{2H} - |t_1 - t_2|^{2H}\right), \quad t_1, t_2 \geq 0.$$

Since the autocovariance of this process is homogeneous of degree $2H$ (for positive scalars), the fBm is self-similar (Definition 2.4). Additionally, by the continuous-time analogue of Definition 2.9, the process is stable (specifically, it is *symmetric α-stable*, see Definition A.27), since the linear combination $aB_{H1}(t) + bB_{H2}(t),\ a, b > 0$ of two independent fBms with autocovariances $\sigma_1^2\rho_B$ and $\sigma_2^2\rho_B$, respectively, is itself an fBm with autocovariance $\left((a\sigma_1)^2 + (b\sigma_2)^2\right)\rho_B$.

In order to determine how to generate discrete samples of a fractional Brownian motion, consider the discrete incremental process $\Delta B_{H;0}, \Delta B_{H;1}, \ldots$ given by

$$\Delta B_{H;n} = B_H(n) - B_H(n-1), \quad n = 1, 2, \ldots \tag{3.1}$$

with $\Delta B_{H;0} = B_H(0)$. This process is called *fractional Gaussian noise* (abbr. fGn). It is clear from (3.1) that $B_H(t) = \sum_{k=0}^{t} \Delta B_{H;k}, \quad t \in \mathbb{N}$ and that $\Delta B_{H;n}$ is Gaussian with zero-mean. The autocovariance of $\Delta B_{H;n}$, however, is slightly more involved. We have

$$\begin{aligned}
\operatorname{Cov}\left[\Delta B_{H;n+k}, \Delta B_{H;n}\right] &= \mathbb{E}\left[\left(B_H(n+k) - B_H(n+k-1)\right)\left(B_H(n) - B_H(n-1)\right)\right] \\
&= \rho_B(n+k, n) - \rho_B(n+k, n-1) - \rho_B(n+k-1, t) + \rho_B(n+k-1, n-1) \\
&= \frac{1}{2}\left(|n-(n+k)|^{2H} - |(n-1)-(n+k)|^{2H} - |n-(n+k-1)|^{2H} + |(n-1)-(n+k-1)|^{2H}\right) \\
&= \frac{1}{2}\left(-2|k|^{2H} + |k+1|^{2H} + |k-1|^{2H}\right), \quad k \in \mathbb{Z},
\end{aligned}$$

which depends on a single parameter k, implying that (3.1) is wide-sense stationary (see Definition A.26).

Leveraging that the covariance of $\Delta B_{H;n}$ depends only on the distance k, we choose to define the autocorrelation sequence as

$$\rho_{B;k} = \mathbb{E}\left[\Delta B_{H;n}\Delta B_{H;n+k}\right], \quad k \in \mathbb{Z}, \tag{3.2}$$

in which the value of $n > 0$ used is arbitrary.

[Hosking, 1984] provides an algorithm for recursively simulating samples $x_0, x_1, \ldots, x_N$ of a stationary process with a given ACS $\rho_{X;k}$ and a normal marginal distribution. By using this method to simulate the fGn given by (3.1), the kth entry of an equidistantly spaced sampling of the fBm $B_H(t)$ can be obtained as the cumulative sum of the first k entries of the simulated fGn.

The algorithm, as described in [Hosking, 1984, Algorithm A], is as follows: First, a starting value x_0 is generated from the distribution $\mathcal{N}(0, \sigma_0^2)$, where σ_0^2 is the stationary variance chosen for the x_n. We define $N_0 = 0$ and $D_0 = 1$ and let ρ_n be defined by (3.2). Next, for $n = 1, \dots, N-1$, the x_n are recursively generated by first computing the means m_n and variances σ_n^2 of the marginal distributions

$$\begin{aligned}
\gamma_n &= \rho_n - \sum_{k=1}^{n-1} \phi_{n-1,k}\rho_{n-k}, \\
D_n &= D_{n-1} - \frac{\gamma_{n-1}^2}{D_{n-1}}, \\
\phi_{n,n} &= \frac{\gamma_n}{D_n}, \quad \phi_{n,k} = \phi_{n-1,k} - \phi_{n,n}\phi_{n-1,n-k}, \quad k = 1, \dots, n-1, \\
m_n &= \sum_{k=1}^{n} \phi_{n,k}x_{n-k}, \quad \sigma_n^2 = \left(1 - \phi_{n,n}^2\right)\sigma_{n-1}^2,
\end{aligned}$$

from which x_n can be generated from the distribution $\mathcal{N}(m_n, \sigma_n^2)$.

Hosking's method is rather slow. It is of complexity N^2, whereas Davies and Harte's method - an alternative to Hosking's method described in [Davies and Harte, 1987] - has complexity $N\log(N)$ [Dieker and Mandjes, 2003]. However, the Davies Harte method does not work when H is close to 1, while Hosking's method simulates the required process exactly regardless of the choice of H.

3.2.2 Designing the Neural Network

In order to assess whether a deep learning-based approach yields any benefit over the wavelet-based method when it comes to estimating the Hurst exponent of a fractal time series, a convolutional neural network (See Appendix C.2) for Hurst exponent estimation of fractional Brownian motion was designed. It was implemented in the Python 3.7, using the `keras` module included in the `tensorflow` package.

The dataset used for training, validation and testing consisted of 2^15 discretely sampled standardized fractional Brownian motions, each of length 2^{14} and with Hurst exponents drawn uniformly from the open interval $(0,1)$. Among the 2^{15} signals, 75% were used for training, 12.5% were used for validation between training epochs, and 12.5% were used for testing after training had concluded. The network was intended to yield a single output, corresponding to the Hurst parameter estimate, and the network was provided with the true Hurst parameters as labels for the training.

A convolutional neural network is a network with at least one convolutional layer. Whereas fully connected layers assign weights between every input node and every output node, all of which must be adjusted through training, convolutional layers convolves the inputs with a set of kernels, whose weights are adjusted through training. This means that convolutional layers are suitable for problems in which there is spatial of temporal correlation between neighboring inputs; such is the case for this thesis.

Pooling functions are often used in combination with convolutional layers. The most notable pooling function is max pooling, which divides the signal into sections of equal size and outputs the maximum value of each section. Pooling layers are used to reduce the dimension of the data, and are often inserted following a convolutional layer; the concolutional layer extracts local features across the inputs, and the max pooling layer passes the most significant values to the next layer, discarding the rest.

The goal of designing the network was to obtain a model that based on an input signal of length 2^{14} outputs a single parameter, the Hurst exponent. This is a problem of regression, in which the output is between 0 and 1. For this reason, the logistic sigmoid, which outputs values between 0 and 1 was used as the output activation function. A number of different networks were designed for this task prior to arriving at the final model.

Sequential networks with fully connected layers were quickly ruled out; the loss function never came close to the MSE observed for the wavelet-based method, and the training of the network was time-consuming, due to the abundance of weights that required training. Sequential implementations of convolutional neural networks were also attempted; they achieved better results, albeit not at the level of the wavelet-based method.

Inspired in part by the Daubechies wavelet decomposition, the neural network used in this thesis applies convolutional layers with varying kernel sizes to the data in parallel. The convolutional layers filter the data and the max pooling layers reduce the data in size; the combination of the two operates in a manner similar to that of the Daubechies wavelet decomposition. Thus, by running the convolutional layers and max poolings in parallel with various kernel sizes and pool sizes, the network is able to observe the data at different scales. The observations at different scales can later be concatenated and processed by a fully connected layer, in order to extract relevant features. The network is depicted in Figure 3.2; two variants of the same general model are included, with one including four parallel segments, and a smaller variant including three parallel segments. The two variants will be referred to by the number of 'scales' they each consider; the larger network will be called the *4-scales CNN* or *CNN4* and the smaller network will be called the *3-scales CNN* or *CNN3*.

For the training of the two models, the stochastic gradient descent (see Appendix C.1.2) was used with the mean squared error as loss function and a batch size of 16, meaning that the weights of the network were iteratively updated in the opposite gradient of the MSE computed on 16 time series per iteration. The network was trained on the data for a total of 200 epochs - in other words, the entire set of training data was cycled through 200 times.

Dropout layers - which randomly excludes outputs at a rate 0.3 during the training - were included after each max pooling layer convolutional layers to avoid overfitting. Batch normalization layers were included at the input and between the fully connected layer and the output. To avoid overfitting the model, the weights used in the test were chosen as to correspond roughly to the epoch with the minimum MSE of the validation data. The MSE is plotted against the epochs in **??**, from which it appears that the validation error generally decreases along with the training error throughout the duration of the training, which suggests that the model is not

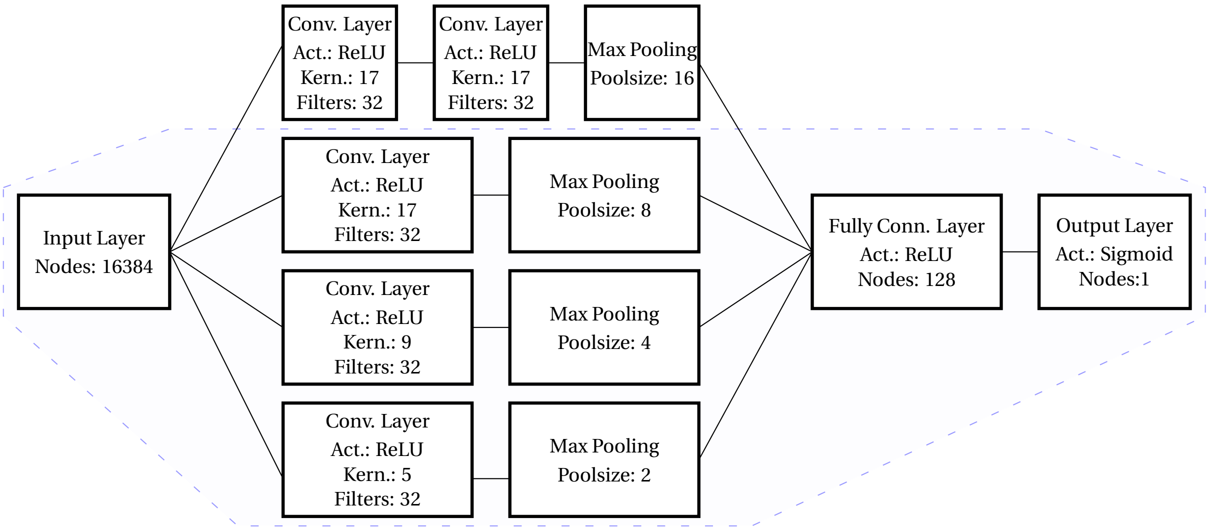


Figure 3.2. A model depicting the structures of the two parallel CNNs designed in this section. The full model depicts the structure of the 4-scales CNN, while the elements marked with blue are the layers included in the 3-scales CNN

overfitted.

3.2.3 Performance Comparison with Wavelet-Based Estimator

In order to assess the performance of both networks designed in this section and the wavelet-based Hurst estimator, a new set of 2^{16} realizations of fBm, each of length 2^{16}, were generated. The models included in this performance analysis include both the 3-scale CNN and 4-scale CNN from the previous subsection, as well as the two variants of the wavelet-based

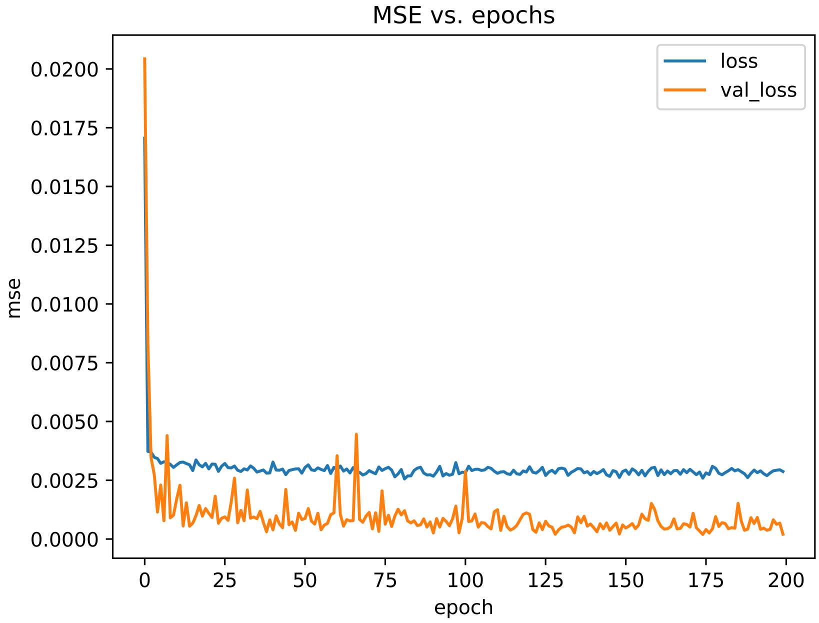


Figure 3.3. MSE of the 4-scales CNN across epochs of the training. The gap between training loss and validation loss is likely due to dropout being active during training only.

Hurst exponent estimator, previously referred to as the average-log and average-square models. The MSE and bias was computed for each model. To gain further insight into the model performances, the MSE was computed on four smaller intervals of the true Hurst exponent; $0 < H < 0.2$, $0 < H < 0.4$, $0.6 < H < 1$ and $0.8 < H < 1$. Hurst exponents close to zero have been known to cause issues for the wavelet-based estimators [Wu, 2020].

The results of the performance analysis are given in Table 3.3

Table 3.3. The bias and MSE of each estimator in the performance analysis of fBm Hurst estimation. The MSE for specific ranges is also given. The marked cells indicate the best performance in each category.

$\hat{H}$	Bias	MSE	MSE(H<0.2)	MSE(H<0.4)	MSE(H>0.6)	MSE(H>0.8)
CNN3	2.667×10^{-2}	1.682×10^{-3}	1.580×10^{-3}	9.746×10^{-4}	2.880×10^{-3}	1.624×10^{-3}
CNN4	3.121×10^{-3}	2.036×10^{-4}	2.124×10^{-4}	1.233×10^{-4}	3.578×10^{-4}	4.431×10^{-4}
$\hat{H}_Y$	-3.427×10^{-2}	3.842×10^{-3}	1.557×10^{-2}	8.176×10^{-3}	1.112×10^{-3}	1.254×10^{-3}
$\hat{H}_V$	-5.753×10^{-3}	1.329×10^{-3}	5.090×10^{-3}	2.899×10^{-3}	2.899×10^{-3}	3.108×10^{-4}

The 4-scales parallel CNN model outperforms the other models in all categories except the estimation of Hurst exponents close to 1, where the average-square variant of the wavelet-based estimator achieved a lower MSE.

Interestingly, the results suggest that while the performance of the wavelet-based estimator decreases as the Hurst exponent decreases, the CNN-based method does not experience this problem; in fact it performs better at the lowest range than at the highest range. One possible explanation for this

Admittedly, the comparison of the two methods is not entirely fair, since the superior performance of the CNN could also be explained by the fact that the data used for this test is statistically identical to its training data. It is entirely possible that, if given a self-similar process with a covariance structure different from that of fBm, the output of the network would convey no meaningful information about the self-similarity of the input process. Regardless, the performance results of this chapter indicates that

3.3 Simulation of EEG Microstate Signals

When developing methods for signal processing, synthetic data is often a useful tool for data-driven learning in cases where a large amount of data is difficult to obtain or when labels for the data are impossible to obtain. Additionally, for synthetic data, the true parameters of the simulated model are readily available, and the model can be adjusted if needed. In order to generate synthetic EEG data, a mathematical model that mimics the behavior of EEG signals reasonably well is required. The aim of this section is to construct such a model.

Recall (1.4), which expressed the fundamental assumption of the microstate model and was given as

$$\boldsymbol{x}_n = \sum_{i=1}^{K} a_{i,n}\boldsymbol{m}_i + \boldsymbol{v}_n, \quad n \in \mathbb{Z},$$

with $K \in \mathbb{N}$ denoting the number of microstate classes, $a_{i,n} \in \mathbb{R}$ denoting the intensity of the ith microstate class at time n, $\boldsymbol{m}_i \in \mathbb{R}^{N_s}$ denoting the ith microstate class template and $\boldsymbol{v}_n \in \mathbb{R}^{N_s}$ being zero-mean additive noise. Furthermore, in order for the microstates be be non-overlapping with one active microstate at a time, we assume

$$\begin{aligned} a_{i,n}a_{j,n} &= 0, \quad \forall i \neq j, \\ \sum_{i=1}^{K} a_{i,n}^2 &> 0, \quad \forall n. \end{aligned}$$

By deciding on a set of K microstate class templates $\boldsymbol{m}_i$ and making some assumptions on the statistics of the intensities $a_{i,n}$ and the noise process $\boldsymbol{v}_n$, the model could be used to generate synthetic microstate data with relative ease. Depending on which assumptions are made, this model may only represent a very idealized subset of possible EEG recordings and might be inadequate in applications where the data needs to be representative of real-world data, as can be the case for data-driven learning. Regardless, the model is consistent with the modified K-means algorithm and should be sufficient for demonstrative use.

A natural choice for the microstate class templates is to either take inspiration from the canonical microstate classes from Section 1.2.1 or simply import the results from a microstate estimation performed on real EEG data. Since the real-world nature of the noise process $\boldsymbol{v}_n$ is largely unknown, modeling the noise process as a set of iid. N_s-variate Gaussian distributed random vectors seems a natural choice. This results in a model depicting an idealized case, since this assumption neglects to account for artifacts present in real EEG recordings, such as eye movements, which are clearly non-Gaussian in nature, causing heavy spikes in recordings, disproportionally targeting anterior sensors and spanning several samples. However, since artifacts are removed through preprocessing of the signal, their absence should not be an issue.

Further assumptions on the intensity parameters $a_{i,n}$ are now all that remains for the model to function as a basis for synthetic data generation. First we assume that the intensity of the signal is independent of which microstate is active. This means noticeable jumps between two subsequent samples will occur whenever the model transitions from one microstate to the next, but this is an inevitability, given the assumption of non-overlapping microstates. Now, by introducing the total intensity process α_n and the class label, $c_n \in \{1, 2, \ldots, K\}$, we can make the substitution $\sum_{i=1}^{K} a_{i,n}\boldsymbol{m}_i = \alpha_n \boldsymbol{m}_{c_n}$. This yields the model

$$\boldsymbol{x}_n = \alpha_n \boldsymbol{m}_{c_n} + \boldsymbol{v}_n, \quad n \in \mathbb{Z}, \tag{3.3}$$

based upon which we may decide on the simulation of the intensity process α_n and the simulation of the sequence of microstate classes separately.

One significant benefit of separating the microstate class from the intensity parameters is that the microstate sequence $(c_n)_{n\in\mathbb{Z}}$ consisting of the class labels, can be treated separately from the rest of the model. Throughout the rest of this section, it is sufficient to assume that c_n is a sequence of elements from the set $\{1, 2, \dots, K\}$ that is "piece-wise constant" in the sense that c_n cycles between intervals within which its value is constant, with the length of these intervals corresponding to the duration of each microstate.

3.3.1 Modeling of the Intensity Process

Deciding on a sensible choice for how the sequence α_n should be generated is perhaps the biggest challenge of generating simulated EEG signals. The polarity of EEG microstates has been found to repeatedly invert [Lehmann, 1971], which suggests that α_n occasionally switches polarity in a periodic manner. When examining fluctuations in EEG recordings of alpha-wave activity (activity in the frequency range of $7.5 - 13$ [Hz]), [Lehmann, 1971] found that successive peaks of inverted polarity generally appeared at intervals of approximately 50 [ms], which corresponds to half of one alpha cycle. This suggests that α_n should be modeled as a 100[ms]-periodic function with varying amplitude.

Before delving into how to model the function, it might be illuminating to examine how large the peaks of α_n should be, by observing real EEG data. Since peaks in the GFP (Definition 1.1) are known to correspond to the points of highest SNR [Lehmann et al., 1987] and since we have assumed the noise process $\boldsymbol{v}_n$ of (3.3) to be independent of α_n and c_n, the process $\boldsymbol{v}_n$ may be considered negligible when observing peaks in the GFP. From the GFP of an average-referenced data point $\boldsymbol{x}_n$, we see that

$$\begin{aligned}
\mathrm{GFP}_n &= \sqrt{\frac{1}{N_s}\sum_{i=0}^{N_s-1}\left(x_{n;i}\right)^2} \\
&\approx \sqrt{\frac{1}{N_s}\sum_{i=0}^{N_s-1}\left(\alpha_n m_{c_n;i}\right)^2} \\
&= \frac{|\alpha_n|}{\sqrt{N_s}}\sqrt{\sum_{i=0}^{N_s-1}\left(m_{c_n;i}\right)^2} \\
&= \frac{|\alpha_n|}{\sqrt{N_s}}, \qquad (3.4)
\end{aligned}$$

where the last equation follow from the microstate template vectors $\boldsymbol{m}_k$ being normalized.

The result (3.5) indicates that the peaks of the GFP scaled by a factor $\sqrt{N_s}$ can be used as a basis for deciding the absolute magnitude of the peaks of α_n. Note that this only applies to the peaks of the GFP, where the noise process can be considered negligible. Additionally, since the studies used in this section to justify the model concerned alpha band activity ($8 - 13$ [Hz]), their data was bandpass-filtered as to attenuate other frequency bands. Thus, the real EEG data used to determine peak magnitudes for this model should be bandpass-filtered as well. While the EEG microstate model is generally used to account for a broader bandwidth of EEG data in more recent studies (e.g. [Van De Ville et al., 2010],[Al Zoubi et al., 2019]), restricting the model to alpha band activity allows for easier simulation.

A simple model that allows for simulation of a periodic function of varying amplitude is one which consists of a sinusoid function with time varying amplitudes. When modeling alpha band EEG signals, it can be assumed that the peaks of α_n are spaced at distances k_p given as half the length of one alpha-wave period, i.e. $k_p \approx 50$ [ms]. Choosing to simulate EEG signals recorded at a sample rate f_s (in [Hz]) implies that the distance between successive peaks in samples should be $k_p = f_s k_p$, which may be rounded to the nearest integer for convenience. Consider the model given as

$$\alpha_n = \beta_n \cos\left(\frac{\pi}{k_p} n\right),$$

which is a sinusoid with an amplitude modulation defined by the discrete process β_n. The purpose of β_n is to model the transition from the peak at time $i k_p$ to the peak at time $(i+1)k_p$ for $i \in \mathbb{Z}$. To avoid introducing additional peaks, β_n should be monotonous between peaks. Denoting the peak at time $n = 0$ by $y[0]$ and neighboring peaks at time $n = i k_p$ by y_i, this β_n can be modeled by means of the piece-wise affine model given as

$$\beta_n = y_{\lfloor n/k_p \rfloor} + \frac{y_{\lfloor n/k_p \rfloor + 1} - y_{\lfloor n/k_p \rfloor}}{k_p} n, \quad n \in \mathbb{Z},$$

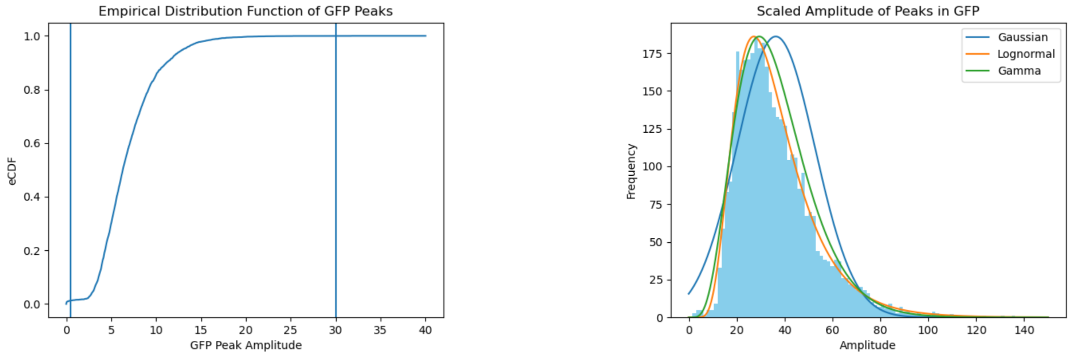
with $\lfloor n/k_p \rfloor$ denoting the integer division of n by k_p. All that remains now before the final model can be obtained is to assess how to model the peak amplitudes y_n should be modeled by using **??**. This will be the focus of the following subsection.

3.3.2 Estimation of GFP Peak Amplitude Distribution

Before the final model can be stated, the statistical properties of the varying amplitudes of the GFP of real EEG data should be examined in order to provide a bassi for deciding on the amplitude of the peaks of α_n.

The sample data used for the estimation was recorded at 512 [Hz] using $N_s = 27$ electrodes. It is of length 29323 in samples, corresponding to approximately 57 seconds. The data contains both a wildly oscillating segment in the beginning of the recording and multiple segments of inactivity. While the first 2000 samples were removed to account for the former abnormality, the data was not trimmed further to account for inactive segments, as the GFP peaks resulting from these segments were assumed to be easily rejected as outliers. Using the built-in functions of EEG lab [Poulsen et al., 2018], the data was first downsampled to 128 [Hz] and subsequently bandpass-filtered using a FIR filter with cutoff frequencies at 5 and 15 [Hz] to remove frequency content outside the alpha band (8 – 13 [Hz]).

The filtered data was exported and analyzed in Python 3.8. The GFP was first computed. Peaks were found using the `find_peaks` function from the `signal` subpackage of `scipy`. GFP values outside the range $[0.5, 30]$ were excluded from the set of GFP peaks. While GFP peaks < 0.5 accounted for approximately 1% of the peaks, these peaks were located exclusively within segments of inactivity. GFP peaks > 30 accounted for only 2 cases among the 3969 peaks in the GFP and were classified as extreme outliers. These point are indicated by vertical lines over the estimated cumulative density function in Figure 3.4a. After removing outliers in the data, the data was scaled by $\sqrt{N_s}$ in accordance with (3.4). A histogram depicting the distribution of the scaled amplitude of the GFP peaks is shown in Figure 3.4b. The GFP peak amplitudes are non-negative and appear to exhibit skewness. Examples of distribution models that exhibit these characteristics include the lognormal distribution Definition A.21 and gamma distribution Definition A.22. These distributions have been fitted to the set of scaled GFP peak amplitudes using the `stats` subpackage of `scipy`. They are shown in Figure 3.4b along with the Gaussian distribution, which is used to provide a baseline.



(a) The empirical cumulative density function for the GFP peak amplitudes prior to scaling and removal of outliers.

(b) Histogram of scaled GFP peak amplitudes with outliers removed. The bin width is 1.5. Fitted probability density functions are scaled to match the max bin height and overlaid for illustrative purposes.

Figure 3.4. Plots of empirical cumulative density function and histogram of GFP peaks.

Section 3.3.2 show Q-Q plots for each of the fitted models, with the theoretical quantiles of the fitted model on the x-axis being plotted against the corresponding quantiles of the set of samples on the y-axis. The red lines indicate the optimal fit, where the quantiles of the theoretical distribution align perfectly with those of the samples. From these plots, it appears the models either underestimate or overestimate the most extreme quantiles of the observed data. The Kolmogorov-Smirnov test can be used to give a more substantiated indication of which model provides the better fit.

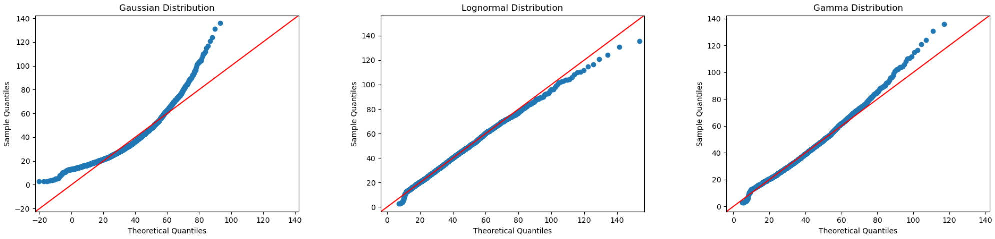


Figure 3.5. Q-Q plots of the fitted Gaussian, lognormal and gamma distributions.

The Kolmogorov-Smirnov test (Definition A.24) provides a simple means of comparing a set of samples to a continuous reference distribution function. It is used to test the hypothesis that the reference distribution is the true distribution of the samples. The Kolmogorov-Smirnov test statistics for the fitted lognormal and gamma distributions were 0.0131 and 0.0325, respectively. This corresponds to a p-value of 0.5038 for the lognormal model and 0.0005 for the gamma model, which indicates that the latter model is clearly rejected on the 0.05-level. Thus, among these options, a lognormal distribution seems to provide the best model for modeling peaks in the amplitude of GFP and, by extension, peaks in the process $|\alpha_n|$.

The assumption that GFP peak sizes are independent is unlikely to hold for real EEG data. It seems probable that neighboring GFP peaks should be correlated, since the GFP quantifies the level of brain activity. The level of brain activity is likely to change relatively slowly, as it depends on factors such as the task the brain is engaged in or the mental state of the subject. However, drawing each peak amplitude independently provides a much simpler model than one that takes correlation between into account, making for easier simulation, especially for signals of large length.

3.3.3 Final Model and Summary

The final model for generating synthetic EEG data can be summarized as follows: Given a set of K different microstate classes $\boldsymbol{m}_k \in \mathbb{R}^{N_s}$ $k \in \mathbb{N}$ and a sequence of integers $c_n \in \{1,2,\ldots,K\}$ indicating the microstate class active at time n, a synthetic EEG signal $\boldsymbol{x}_n, n \in \mathbb{Z}$ is obtained from

$$\boldsymbol{x}_n = \alpha_n \boldsymbol{m}_{c_n} + \boldsymbol{v}_n, \quad n \in \mathbb{Z}, \tag{3.5}$$

where $v_n \overset{iid.}{\sim} \mathcal{N}(0,\sigma_v^2)$ and

$$\alpha_n = \beta_n \cos\left(\frac{\pi}{k_p} n\right), \quad n \in \mathbb{Z}$$

in which k_p denotes the chosen distance between successive peaks of α in samples and

$$\beta_n = Y_{\lfloor n/k_p \rfloor} + \frac{Y_{\lfloor n/k_p \rfloor + 1} - Y_{\lfloor n/k_p \rfloor}}{k_p} n, \quad n \in \mathbb{Z}$$

with $\lfloor n/k_p \rfloor$ denoting the integer division of n by k_p and $Y_k \overset{iid.}{\sim} \log\mathcal{N}(\mu_y,\sigma_y^2), k \in \mathbb{Z}$.

Ideally, by letting f_s denote the sampling frequency (in [Hz]) and t_p denote the time (in [s]) between successive peaks, k_p should be chosen as to satisfy $k_p = f_s t_p$. However, this is very likely to yield a non-integer value for k_p, which means that the peaks are never guaranteed to be exactly $\pm\beta_n$. Choosing $f_s = 512$ and recalling that $t_p = 0.05$ for alpha band activity (10 [Hz]), we have $k_p = 25.6$. Rounded off, we have $k_p = 26$.

The specific parameters used for the model are given in Table 3.4.

Table 3.4. The parameters chosen for the model (3.5).

Parameter	Value
σ_v^2	10
μ_y	3.5
σ_y^2	0.20
k_p	26

3.3.4 Microstate Estimation and Classification on Simulated EEG Signal

To test the synthetic data model introduced in this section, some of the features of the synthesized signals were investigated in EEGLAB.

First, a set of example microstates prototypes were imported to be used as a template for the synthetic data; they are depicted in Figure 3.6. Next, a set of microstate labels were created to act as synthethic microstate sequences. For this test, the synthetic microstate sequences were deterministic, cycling through each microstate in order, remaining at each for 64 samples. This way, it should be easy to verify the accuracy of the microstate backfitting. The synthetic model was obtained using (3.5) with the parameters given in Table 3.4, and imported into EEGLAB. Channel locations were imported from the same dataset as the template microstate prototypes.

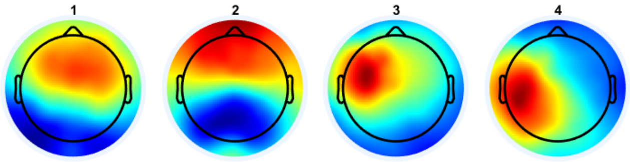


Figure 3.6. The microstate prototypes imported for this test

Figure 3.7 shows a section of the synthesized data along with a section of real EEG data for comparison. While the real EEG data appears natural and stationary, the synthesized data appear to behave much more wildly with several noticeable vertical lines, presumably located at the transitions between microstates. The lack of smoothing of the signal has likely contributed to this behavior; the real signal appears much more correlated.

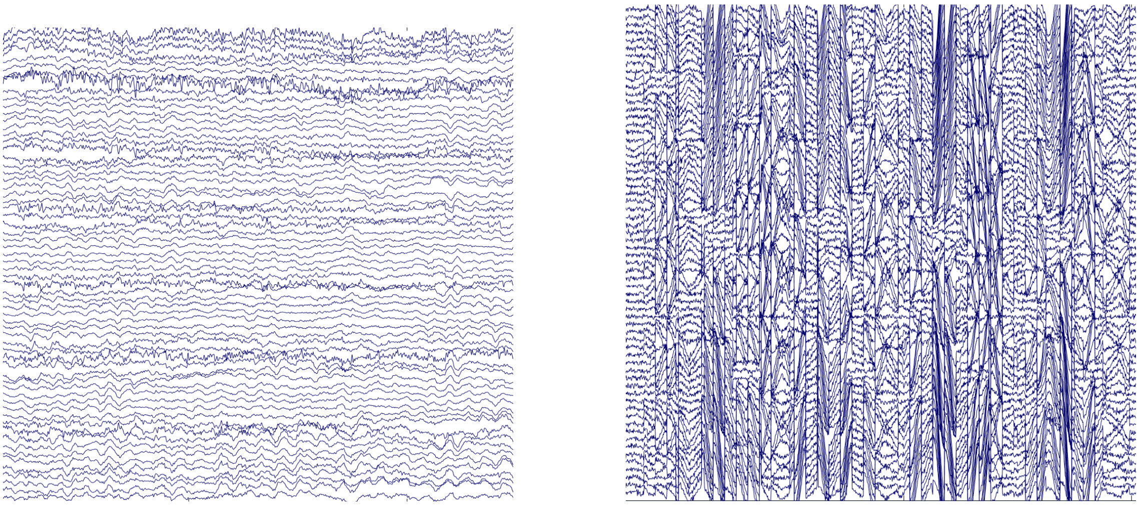
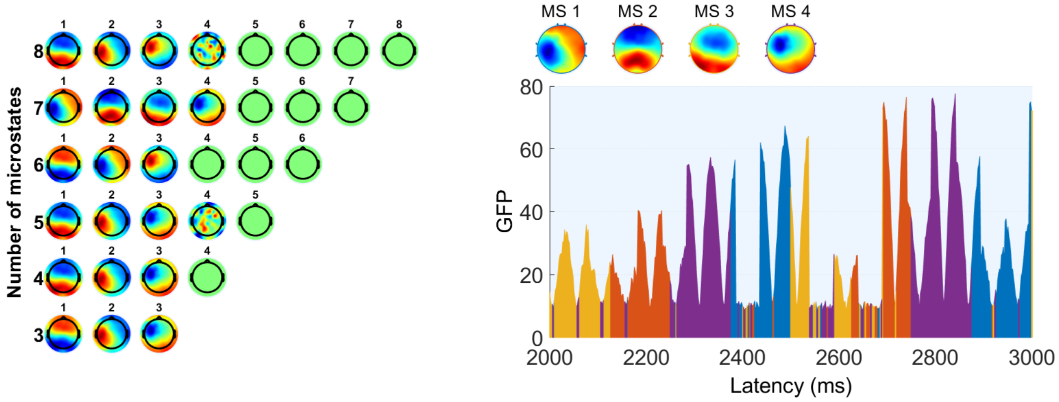


Figure 3.7. Snippets of EEG data scrolls, depicting EEG recordings over time (x-axis) across all channels (y-axis) for real EEG data (left) and synthesized EEG data (right).

The microstate estimates in fig. 3.8a behaved oddly for the synthetic data, with several microstates being estimated. Nonetheless, the backfitting of the microstates, depicted in Figure 3.8b, performed comparatively well. As was to be expected, it produces some errors at low GFP. The jaggedness of the GFP is also quite noticeable in this figure.



(a) The estimated microstate maps for the synthesized data.

(b) Backfitting of the estimated microstates to the synthesized data.

In conclusion, the synthesized model produced mixed results. In terms of providing simple EEG substitutes on which microstate estimation can be performed, it performs reasonably well. In terms of generating EEG data that more closely resembles real EEG, however, it falls short. It is easy to model a sequence in a way that ensures that the right vectors are chosen as microstate prototypes by the mod-K-means algorithm; it is another thing entirely to accurately model the constantly changing and unfamiliar characteristics of neural generators in the brain.

If the model were to be improved upon in the future, a decent starting point would be to include for frequencies, and to potentially introduce some smoothing of the signal, in order to mitigate the jaggedness of the current model. Regardless of the performance of the model, it has provided new insight into some characteristics of EEG signals that was previously left unconsidered.

Analysis of Self-Similarity Parameter Changes due to Acoustic Stimuli 4

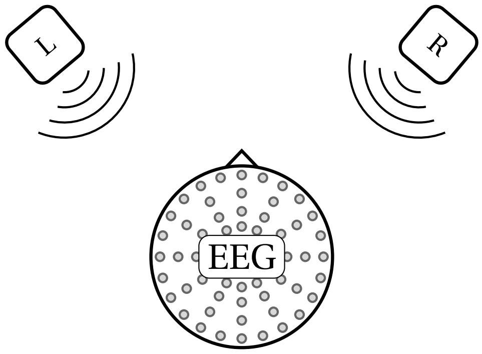
Motivated by recent findings suggesting that EEG microstates are scale-invariant, the aim of this thesis has been to assess whether acoustic stimuli affects the Hurst exponent of random-walk embedded microstate sequences. For this purpose, an analysis has been conducted on EEG microstates obtained from real EEG data. First, the data used in the analysis will be outlined, along with the main properties that will be covered in the analysis.

4.1 Analysis Outline

The data used in this thesis is part of a larger set of data from Simon et al. [2022], and provided by Aalborg University. While the original set of data included 35 subjects, the data is restricted to a single subject for this thesis. The subject, a 24-year old male, was placed in front of two separate speakers playing different sounds simultaneously. The subject was tasked with focusing on one of the sounds while ignoring the other, and their scalp potentials were measured by an EEG with 64 channels. This is illustrated in Figure 4.1

The test consisted of 32 successive trials, covering various combinations of target sound genre, distractor sound genre, and target direction of arrival. The sounds used in the experiment can be divided into two groups, each of which had two subgroups; music signals (either piano pieces or electronic) and speech signals (either a male or a female speaker).

The available data was sampled with a sampling frequency of 512 Hz. In each trial, the target sound starts playing at 2 s, while the distractor sound started playing at 4 s. At 64 s, the target sound stopped, and at 66 s, the distractor sound stopped. In samples, the distance between the start of the target sound and the end of the distractor signal is $64 \cdot 512 = 2^{15}$, which is ideal. Additionally, by splitting the microstate sequences of each trial into two sets, separated at the middle, the Hurst exponent could be estimated twice for each trial, and the CNN introduced in Section 3.2, which was trained on signals of length 2^{14}, could be applied.



D T	M(P)	M(E)	S(F)	S (M)	Σ
M(P)	-	4 (2:2)	1 (1:0)	2 (1:1)	7 (4:3)
M(E)	3 (2:1)	-	2 (0:2)	2 (2:0)	7 (4:3)
S(F)	2 (0:2)	2 (1:1)	-	4 (2:2)	8 (3:5)
S(M)	2 (1:1)	2 (1:1)	4 (1:3)	-	8 (3:5)
Σ	7 (3:4)	8 (4:4)	7 (2:5)	8 (5:3)	30 (14:16)

Figure 4.1. Left: Illustration of the test setup. **Right:** Overview of number of trials within each group. Rows/columns indicate the type of target/distractor signal; music ('M'), piano/electronic ('P'/'E'), and speech ('S'), female/male speaker('F'/'M'). Bracketed numbers indicate the target signal direction of arrival in the indicated group as '(left : right)'.

Naturally, when using any interval that includes either of the 2-4 s or 64-66 s intervals, one assumes that a lack of one of the two signals does not significantly affect the Hurst exponent. To assess whether this is true, the estimated microstate sequences were split into 32 sections of length 2 s in time (2 in samples), and the Hurst exponent was compared between the set of exponents within each group. Previous studies have indicated (e.g. [Wu, 2020]) that while the empirical bias of the wavelet-based estimator is mostly unaffected by sample length, the variance of the estimator increases as the sample length decreases. Thus, when analyzed over all groups, the likelihood that Hurst exponents of boundary sections are distributed identically to those not at the boundaries was obtained.

Most studies on EEG microstates focus on neural activity in the the alpha band (7.5-13 Hz). While this thesis has not specified any one band of frequencies to be of greater interest than any other, a significant portion of the referenced litterature has. For this reason, an additional analysis was performed, in whiuch the set of EEG data was downsampled by a factor 2 and bandpass-filtered to exclude most frequencies outside the alpha band.

The tests mentioned in this section will be elaborated upon in their respective sections.

4.2 Methodology

This section concerns the methodology used for the Hurst analysis. There are a number of steps involved with obtaining Hurst parameter estimates that must be addressed before the analysis itself can be introduced.

The first step (Section 4.2.1) is the preprocessing of the EEG signals; some relevant aspects of preprocessing were originally discussed in Section 1.1.2. This step is followed by the estimation and classification of microstates in the data (Section 4.2.2), a topic originally introduced in Sections 1.2 and 1.3. Next, the microstate sequences obtained from the estimation are embedded into random walks (Section 4.2.3) using the approach introduced in Section 2.3. After obtaining the RW-MS, the Hurst exponents must be estimated for each of the signals (Section 4.2.4), using either the wavelet-based method from Section 2.4.1

4.2.1 Preprocessing

The test data had been preprocessed in advance, having had been uniformly sampled at 512 Hz, average-referenced and divided into separate signals for each trial. Each trial was synchronized such that at sample 1025, the target signal started, and at sample 33793, the distractor signal had ended.

While the data did not appear to require much cleaning in general, the utility frequency at 50 Hz was very prominent when observing the power spectral density, and had to be removed in preprocessing. There were also a number of noisy channels, and indications that some artifacts related to eye or muscle movement were present in the data.

All steps in the preprocessing of the EEG data were performed within EEGLAB, primarily using its built-in functions. The preprocessing scheme of this thesis consists of seven steps for the main analysis; they are summarized here; clarifications on the individual steps follow.

1. Removal of the DC component by highpass-filtering the EEG signals with cutoff at 1 Hz.
2. Removal of the utility frequency at 50 Hz using the `CleanLine` plugin.
3. Rejection of noisy channels using the built-in tools for removing bad channels.
4. Artifact removal using the built-in tools for automatic ICA component classification and removal.
5. Interpolation of rejected channels based on orgininal channel locations.
6. Removal of data outside the 2 - 66 s interval.
7. Manual removal of artifacts based on ICA computed across all data sets.

Steps 2 through 6 are performed on each signal separately; these steps all use automated algorithms with fixed hyperparameters, in order to ensure that the data is processed in a consistent manner, and to streamline the process in a way that minimizes the amount of manual labor required. For steps 1 and 7 the data is processed in a single batch; step 7 is performed manually to account for potential oversights of the automated part of the scheme. Average-referencing was applied prior to steps 1,4 and 7.

The `CleanLine` function used in step 2 seeks to reduce the frequency content near the utility frequency, while leaving the rest of the frequency spectrum unaffected. It may require some parameter adjustments to function as intended. Figure 4.2 shows an example of the effect of applying the function to an EEG signal. Note that the spike at 100 Hz, a harmonic of the utility frequency, remains. While reapplying the `CleanLine` function, the spike at 1000 Hz was deemed negligible

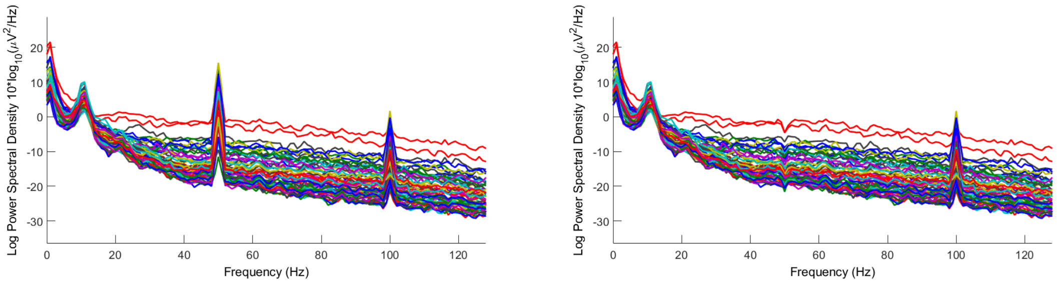


Figure 4.2. Log power spectral densities of EEG channels before and after applying `CleanLine`.

For step 3, the `clean_rawdata` function of EEGLAB iss used to reject channels based on recordings from neighboring channels. Let $x_{n;i}$ denote the EEG recorded at the ith electrode at time n. Based on neighboring electrodes, an estimate $\hat{x}_{n;i}$ can be obtained. If the correlation between $x_{n;i}$ and $\hat{x}_{n;i}$ consistently falls below a given threshold within a length of time, the behavior of the ith channel is considered abnormal enough to warrant rejection.

The ICA and subsequent flagging and removal of artifactual components in step 4 are handled by the built-in functions of EEGLAB. Thresholds were set for rejection of artifacts related to eye movements only; artifacts related the utility frequency or channel noise is assumed to be removed or sufficiently mitigated due to being addressed in the previous steps. An example of automated is given in Figure 4.3, where the topographical map of each independent component is obtained by mapping the corresponding row of the mixing matrix onto the channel locations)

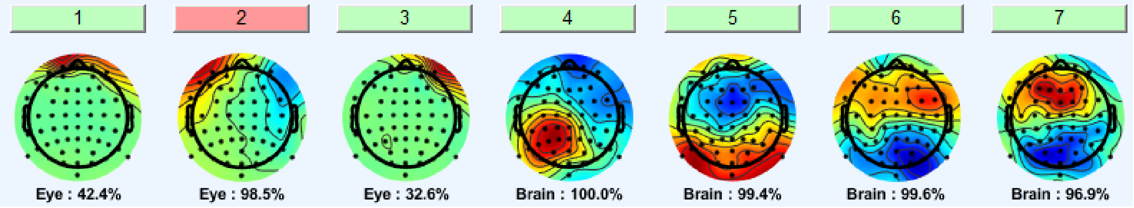


Figure 4.3. The first 7 independent components of an example signal, sorted by their root mean power spectrum. Red and green boxes indicate components to be rejected/accepted, respectively.

Some artifactual components may evade the automated flagging; this is part of the reason step 7 is included in the preprocessing scheme. Figure 4.4 contains further details of the first, second and third components of Figure 4.3, including the power spectral densities and time-frequency spectra of their source signals. Eye artifacts arise ,

4.2.2 Microstate Estimation and Classification

The microstate estimation and classification part of the analysis was performed within EEGLAB, using the `Microstate` toolbox [Poulsen et al., 2018]. After preprocessing, data had to have its

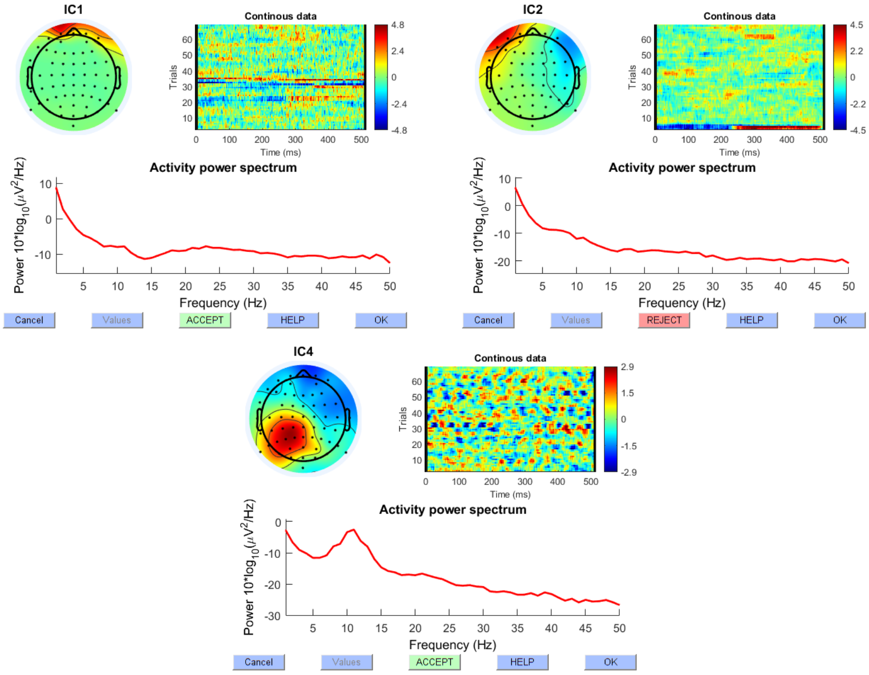


Figure 4.4. Examples of independent components from Figure 4.3, plotted with their power spectral density and time-frequency spectrum. From top-left, clockwise; a potentially artifactual component not flagged as such; a likely artifactual component, flagged as such; a non-artifactual component, likely related to brain activity.

microstates estimated and classified.

The steps included in the microstate classification and estimation were as follows:

1. Concatenation of 1000 data points from each of the 30 trials corresponding to peaks in the GFP.
2. Segmentation of data into K microstates over multiple choices of K.
3. Choice of K based on measures of fit.
4. Backfitting of microstates to each trial.
5. Smoothing of microstate labels.
6. Calculation of Microstate Statistics

Most of the steps listed here have been addressed in earlier sections; step 3 has not. In order to choose the number of microstates for the analysis, EEGLAB includes a number of measures of fit used to describe how well the data is described by the microstates. The three used in this thesis were the *global explained variance*, *cross-validation criterion* and the *dispersion* [Poulsen et al., 2018].

$K = 6$ scored the highest in the measures of fit and was chosen as the number of microstates. $K = 4$ was also included, since it is the amount most popular in the litterature.

Smoothing of the microstate labels were included as fifth step. The effect of smoothing is illustrated on Figure 4.5; by rejecting small segments and redistributing their points, a smoother version of the microstate sequence is obtained.

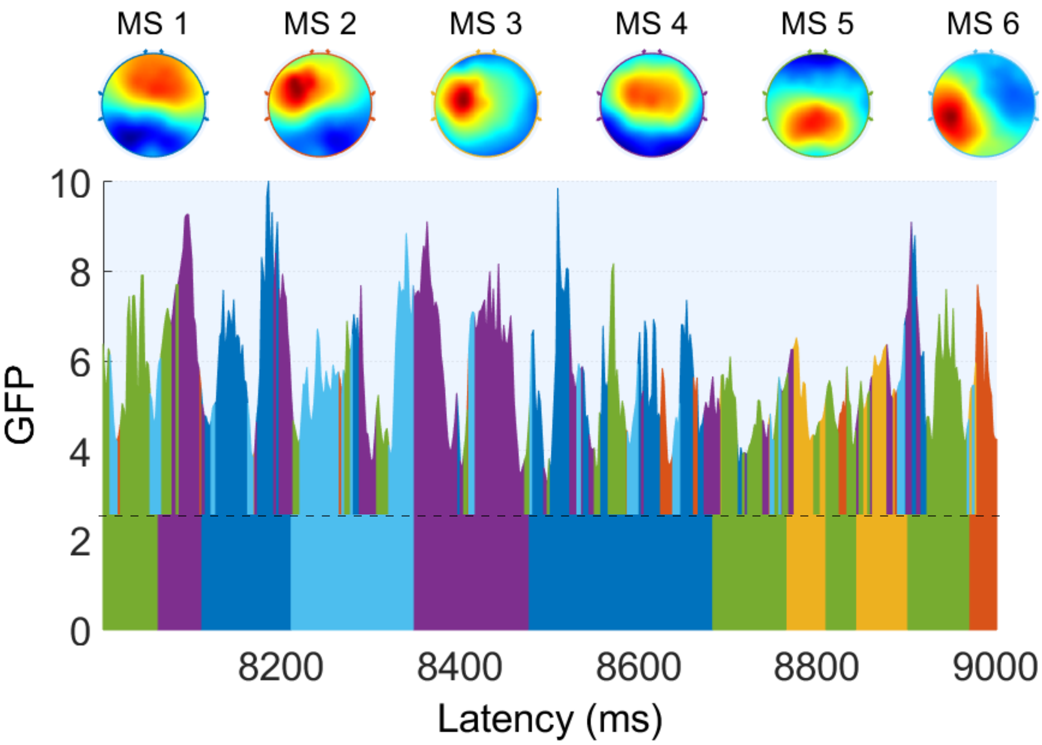


Figure 4.5. An illustrative example of backfitting. The area above the dotted line depicts the backfitted microstates prior to smoothing; the area below the dotted line depicts the smoothed segmentation

In the final step, several microstate statistics are computed. The most notable include the duration, frequency of occurrence and total coverage of every microstate in each signal.

4.2.3 Random Walk Embedding of Microstates

The random-walk embedding of microstates (RW-MS) is applied as per its introduction in Definition 2.7:

$$X_n = \sum_{k=1}^{n} U_k, \quad U_k = \begin{cases} 1, & C_k \in S_1, \\ -1, & C_k \in S_2. \end{cases}$$

The random walk was computed for every possible set of partitions S_1 and S_2 of equal size. The results were passed on to the Hurst exponent estimation.

4.2.4 Hurst Exponent Estimation

The Hurst Exponents were estimated using the wavelet-based estimators $\hat{H}_Y$ and $\hat{H}_V$ summarized in Section 2.4.4 as well as the 4-scales CNN from Section 3.2.

Since no baseline was available to compare against, the outputs of the various Hurst exponent estimators were compared against each other using one-way ANOVA; provided they were in agreement, their estimations were assumed to be reasonable.

4.2.5 Hurst Exponent Analysis

The final step of this analysis was to address the main question asked in this thesis; whether or not changes occur in the Hurst exponent due to acoustic stimuli. The most important factors in this regard were the target signal and distractor signal. If either factor was found to be significant, it would indicate that the type of To examine whether the interaction between the target signal and distractor signal was significant, a two-way ANOVA with an interaction was applied to a linear fit with the factors shown in Figure 4.6, using the `statsmodel` package for Python 3.8.

$$\begin{tikzcd}
& & T \arrow[dr] & \\
I \arrow[r] & T \times D \arrow[ur] \arrow[dr] & & O \\
& & D \arrow[ur] &
\end{tikzcd}$$

Figure 4.6. Diagram of the factors of the two-way ANOVA.

Since the data used in this analysis does not include cases in which both the target and distractor signal are of the same group, the subgroups were concatenated such that there was only one speech groups and music group, each.

The ANOVA was computed from the Hurst estimates from the two wavelet-based models separately. Only a single par of partition sets, S_1 and S_2, were used, being the one corresponding to the most even partition in terms of average probability of $c_n \in S_1$.

4.3 Results

Having described the methodology of the individual steps of the analysis in the previous section, this section aims to summarize the results. The results are divided between three categories.

Section 4.3.1 contains the results obtained from the microstate estimation and classification.

Section 4.3.2 includes observations on the various models applied to the Hurst exponent estimation problem.

Section 4.3.3 contains the main results of this thesis.

4.3.1 Microstates

Figure 4.7 depicts the microstates estimated from the data. They generally appear to be unaffected by noise.

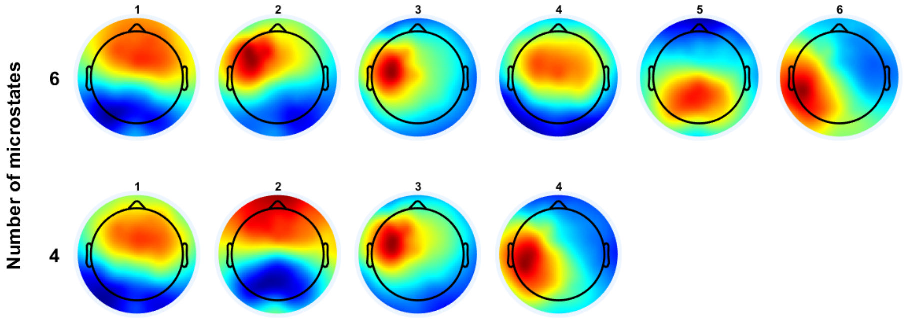


Figure 4.7. The microstates computed for the analsysis.

4.3.2 Hurst Estimation

The wavelet-based Hurst performed well, with one exception; the lower boundary $j_0 = 3$ could not be used in the Hurst estimation of the random walks, since the wavelet coefficients had almost exclusively zero values at low scales. The lower bounds where increased to 5 and 4 for $\hat{H}_Y$ and $\hat{H}_V$, respectively.

The CNN-based estimator generally performed inconsistently and was thus excluded from the analysis.

4.3.3 Hurst Exponent Analysis

The tables below show the ANOVA computed across every Hurst estimate obtained by either of the two wavelet-based methods for Hurst exponent analysis. The results included are the sum of squares, F statistic and their corresponding p values.

Table 4.1. The ANOVA results obtained from the average log Wavelet Hurst Estimator for 6 and 4 microstates, respectively.

Factor	SoS	F	p(>F)	SoS	F	p(>F)
T	0.000	0.002	0.963	6.946	0.022	0.882
D	0.021	5.011	0.029	4.968	0.000	0.999
TxD	0.010	2.493	0.120	2.663	0.009	0.927
Res	0.233			1.753		

Table 4.2. The ANOVA results obtained from the average squares Wavelet Hurst Estimator with 6 and 4 microstates, respectively.

Factor	SoS	F	p(>F)	SoS	F	p(>F)
T	0.011	13.002	0.001	0.012	13.540	0.001
D	0.001	1.595	0.212	0.001	1.396	0.242
TxD	0.000	0.526	0.471	0.000	0.246	0.622
Res	0.048			0.050		

4.4 Discussion

This section will discuss the results of the analysis in order to address the problem statement, which was whether accoustic stimuli affect the Hurst exponent.

The preprocessing, microstate estimation and classification, and RW-MS faced no major issues.

In the estimation of the Hurst exponent that followed, however, both the wavelet-based methods and the CNN-based method ran into some problems. The problems with the former was due to the RW-MS having wavelet coefficients close to zero at a significant portion of the lower range of octaves octaves. This issue was fixed by adjusting boundaries of the observed octaves.

The CNN-based Hurst estimator, however, was not as easily fixed.

In the future, performance could likely be optimized by including more relevant self-similar sequences in the training data. Training a network on various random walks could be a place to start.

As for the main results, of Table 4.1 and Table 4.2, there do appear to be indications that the type of sound signals played are significant, though the specific factor that is implied to be significant changes between the methods used.

Due to time constraints, a portion of the data has not been considered in this thesis. Future work on this topic is needed to either justify or reject the claim that acoustic stimuli affect the Hurst exponent of microstate sequences.

Conclusion 5

The topic of this thesis has been analyzing and assessing changes in the fractal characteristics of EEG microstate sequences due to acoustic stimuli. Specifically, the EEG microstate sequences were embedded into a random walk, so that an estimate of the Hurst exponent could be obtained based on its scaling properties.

The topic of estimating the Hurst exponent played a significant role in this thesis. Two methods were introduced for the purpose of estimating the Hurst exponent; one was an established method based on the wavelet transform; the was through the implementation of a convolutional neural network.

Using the two methods, an ANOVA was performed on EEG data recordings of a listening task to assess whether the type of sound was significant. Ultimately, the results of the analysis proved inconclusive, and further work is needed on the topic.

Bibliography

Abry, P., Delbeke, L., and Flandrin, P. (1999). Wavelet based estimator for the self-similarity parameter of alpha-stable processes. volume 3, pages 1729 – 1732 vol.3.

Al Zoubi, O., Mayeli, A., Tsuchiyagaito, A., Misaki, M., Zotev, V., Refai, H., Paulus, M., Bodurka, J., and the Tulsa 1000 Investigators (2019). EEG microstates temporal dynamics differentiate individuals with mood and anxiety disorders from healthy subjects. *Frontiers in Human Neuroscience*, 13:56.

Binnie, C. D. and Prior, P. F. (1994). Electroencephalography. *Journal of Neurology, Neurosurgery, and Psychiatry*, 57(11):1308–1319.

Bishop, C. M. (2009). *Pattern Recognition and Machine Learning*. Springer. ISBN: 978-0387-31073-2.

Britz, J., Van De Ville, D., and Michel, C. M. (2010). Bold correlates of eeg topography reveal rapid resting-state network dynamics. *NeuroImage*, 52(4):1162–1170.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. John Wiley & Sons, Inc.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996.

Davies, R. and Harte, D. S. (1987). Tests for Hurst effect. *Biometrika*, 74(1):95–101.

Delorme, A. and Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Neurosci Methods*, 134(1):9–21.

Dieker, A. and Mandjes, M. (2003). On spectral simulation of fractional brownian motion. *Probability in the Engineering and Informational Sciences*, 17.

Dinov, M. and Leech, R. (2017). Modeling uncertainties in EEG microstates: Analysis of real and imagined motor movements using probabilistic clustering-driven training of probabilistic neural networks. *Frontiers in Human Neuroscience*, 11:534.

Folland, G. B. (2009). *Fourier Analysis and Its Applications*. American Mathematical Society. ISBN: 978-0-8218-4790-3.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Hosking, J. R. M. (1984). Modeling persistence in hydrological time series using fractional differencing. *Water Resources Research*, 20(12):1898–1908.

Hutchinson, J. E. (1981). Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747.

Jaffard, S. (2004). Wavelet techniques in multifractal analysis. *Proceedings of Symposia in Pure Mathematics*, 72.2:1100–1114.

Jaffard, S., Lashermes, B., and Abry, P. (2007). *Wavelet Leaders in Multifractal Analysis*, pages 201–246.

Jung, T.-P., Makeig, S., Humphries, C., Lee, T.-W., McKeown, M. J., Iragui, V., and Sejnowski, T. J. (2000). Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, 37(2):163–178.

Kay, S. (2006). *Intuitive Probability and Random Processes Using MATLAB*. Springer. ISBN: 978-0-387-24157-9.

Klimesch, W. (1999). EEG alpha and theta oscillations reflect cognitive and memory performance: a review and analysis. *Brain Research Reviews*, 29(2):169–195.

Koenig, T., Prichep, L., Lehmann, D., Merlo, M. C. G., Kochi, K., Hell, D., and Koukkou, M. (1999). A deviant EEG brain microstate in acute, neuroleptic-naive schizophrenics at rest. *European Archives of Psychiatry and Clinical Neuroscience*, 249(4):205–211.

Koenig, T., Prichep, L., Lehmann, D., Sosa, P. V., Braeker, E., Kleinlogel, H., Isenhart, R., and John, E. R. (2002). Millisecond by millisecond, year by year: normative EEG microstates and developmental stages. *NeuroImage*, 16(1):41–48.

Lee, G. R., Gommers, R., Wasilewski, F., Wohlfahrt, K., and O'Leary, A. (2009). PyWavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237.

Lehmann, D. (1971). Multichannel topography of human alpha eeg fields. *Electroencephalography and Clinical Neurophysiology*, 31(5):439–449.

Lehmann, D., Ozaki, H., and Pal, I. (1987). EEG alpha map series: brain micro-states by space-oriented adaptive segmentation. *Electroencephalography and Clinical Neurophysiology*, 67(3):271 – 288.

Lehmann, D., Strik, W., Henggeler, B., Koenig, T., and Koukkou, M. (1998). Brain electric microstates and momentary conscious mind states as building blocks of spontaneous thinking: I. visual imagery and abstract thoughts. *International Journal of Psychophysiology*, 29(1):1–11.

Mallat, S. G. (1989). Multiresolution approximations and wavelet orthonormal bases of L2(R). *Transactions of the American Mathematical Society*, 315(1):69–87.

Mandelbrot, B. B. (1967). How long is the coast of britain? statistical self-similarity and fractional dimension. *Science*, 156(3775):636–638.

Mandelbrot, B. B. (1982). *The fractal geometry of nature*. W.H. Freeman and Co. ISBN: 0-7167-1186-9.

Mishra, A., Englitz, B., and Cohen, M. X. (2020). EEG microstates as a continuous phenomenon. *NeuroImage*, 208:116454.

Murray, M. M., Brunet, D., and Michel, C. M. (2008). Topographic ERP analyses: A step-by-step tutorial review. *Brain Topography*, 20:249 – 264.

Olofsson, P. and Andersson, M. (2012). *Probability, Statistics and Stochastic Processes*. John Wiley & Sons, Inc.

Pascual-Marqui, R. D., Michel, C. M., and Lehmann, D. (1995). Segmentation of brain electrical activity into microstates: model estimation and validation. *IEEE Transactions on Biomedical Engineering*, 42(7):658–665.

Pipiras, V. and Taqqu, M. S. (2017). *Stable Non-Gaussian Self-Similar Processes with Stationary Increments*. Springer, first edition.

Poulsen, A. T., Pedroni, A., Langer, N., and Hansen, L. K. (2018). Microstate EEGlab toolbox: An introductory guide. *bioRxiv*.

Simon, A., Loquet, G., Østergaard, J., and Bech, S. (2022). Auditory attention decoding from eeg during music listening. *Frontiers in Neuroscience*. Submitted for publication.

Simonsen, I., Hansen, A., and Nes, O.-M. (1997). Determination of the hurst exponent by use of wavelet transforms. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 58.

Teplan, M. (2002). Fundamentals of EEG measurement. *Measurement science review*, 2(2):1–11.

Tharwat, A. (2018). Independent component analysis: an introduction. *Applied Computing and Informatics*, ahead-of-print.

Van De Ville, D., Britz, J., and Michel, C. M. (2010). EEG microstate sequences in healthy humans at rest reveal scale-free dynamics. *Proc Natl Acad Sci*, 107(42):18179–18184.

Veitch, D. and Abry, P. (1999). A wavelet-based joint estimator of the parameters of long-range dependence. *IEEE Transactions on Information Theory*, 45(3):878–897.

von Wegner, F., Tagliazucchi, E., and Laufs, H. (2017). Information-theoretical analysis of resting state EEG microstate sequences - non-Markovianity, non-stationarity and periodicities. *NeuroImage*, 158:99–111.

Wendt, H., Abry, P., and Jaffard, S. (2007). Bootstrap for empirical multifractal analysis. *IEEE Signal Processing Magazine*, 24(4):38–48.

Wu, L. (2020). A note on wavelet-based estimator of the hurst parameter. *Entropy*, 22(3).

Definitions and Theorems A

This chapter contains relevant definitions and results not introduced in the main chapters of the report. As this chapter is intended only as a reference work, the following sections are only loosely structured.

A.1 Set Theory and Functions on $\mathbb{R}^n$

Definition A.1 (L^2-norm)
The L^2-norm of a function $f : \mathbb{R} \to \mathbb{C}$ is defined as

$$\|f\|_2 = \sqrt{\int_{-\infty}^{\infty} |f(t)|^2 dt} \qquad \blacktriangle$$

Definition A.2 (Square-Integrable Function)
A square-integrable function (or L^2-function) is a function $f : \mathbb{R} \to \mathbb{C}$ for which the integral of the square of the absolute value of f is finite, i.e.

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty.$$

The set of all square-integrable functions on $\mathbb{R}$ is denoted by $L^2(\mathbb{R})$. ▲

A product of two square-integrable functions $f(t)$ and $g(t)$ is also integrable. Since $||f(t)||^2 + ||g(t)||^2 - 2|f(t)||g(t)| = (|f(t)| - |g(t)|)^2 \leq 0$ for all t, it follows that $|f(t)g(t)| \leq \frac{1}{2}(|f(t)|^2 + |g(t)|^2)$ for all t. Since $|f(t)|^2 + |g(t)|^2$ acts as an upper bound for $|f(t)g(t)|$ and by Definition A.2 has a bounded integral, the product $|f(t)g(t)|$ must also have a bonded integral [Folland, 2009, p. 74].

Definition A.3 (Diameter of a Set)
The diameter of a set $S \subset \mathbb{R}^n$ is the largest distance between any two points in the set, i.e.

$$\operatorname{diam}(S) = \sup_{\boldsymbol{x},\boldsymbol{y} \in S} \left(\|y - x\|\right),$$

where $\|\cdot\|$ is the L^2-norm Definition A.1. ▲

Definition A.4 (Square-Summable Sequence)

A square-summable sequence (or l^2-sequence) is a sequence $\{x_i \in \mathbb{C}^n : i \in \mathbb{Z}^k\}$ for which the sum of the squares of the absolute values of each entry of x_i is finite, i.e.

$$\sum_{i \in \mathbb{Z}^k} |x_i|^2 dx < \infty.$$

The set of all square-summable sequences on $\mathbb{Z}^k$ is denoted $l^2(\mathbb{Z}^k)$. ▲

Definition A.5 (Support)

The *support* of a function $f : \mathbb{R}^n \to \mathbb{C}$ is the closure of the set of all points for which $f(\boldsymbol{x}) \neq 0$ [Folland, 2009, p. 304]. In other words,

$$\text{supp}(f) = \overline{\{\boldsymbol{x} : f(\boldsymbol{x}) \neq 0\}},$$

in which the overline is used to denote the closure of the set, i.e. $\overline{S}$ is the smallest closed set containing S. If $\text{supp}(f) \subseteq S$, we say that f is supported in S. ▲

Definition A.6 (Compact Subset)

A *compact subset* of $\mathbb{R}^n$ is a subset $S \subset \mathbb{R}^n$ which is closed and bounded. ▲

Definition A.7 (Compact Support)

A function $f : \mathbb{R}^n \to \mathbb{C}$ has *compact support* if its support $\text{supp}(f)$ is a compact subset of $\mathbb{R}^n$. ▲

Definition A.8 (Isometry)

Let X and Y be metric spaces with metrics d_X and d_Y. A map $f : X \to Y$ is called an isometry if it preserves distance between any two a and b from X. More formally, f is called an isometry if

$$d_Y(f(a), f(b)) = d_X(a, b), \quad \forall a, b \in X.$$ ▲

Definition A.9 (Orthonormal Family)

An orthonormal family of functions on $\mathbb{R}$ is a set of N square-integrable functions $f_i, i = 1, 2, \ldots, N$ which are orthogonal to one another and of unit length. In other words,

$$\int_{-\infty}^{\infty} f_i(t) f_j^*(t) dt = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}, \quad 1 \leq i, j \leq N$$ ▲

Definition A.10 (Orthonormal Basis of L^2)

An orthonormal basis of $L^2(\mathbb{R})$ is an orthonormal family (Definition A.9) that spans $L^2(\mathbb{R})$. ▲

Definition A.11 (Dense Set)

We say a set $S \subset X$ is *dense* in X if its closure constitutes the whole set X, i.e. if every point in X either belongs to S or is a limit point of S. ▲

Definition A.12 (Contraction Mapping)

A contraction mapping on $\mathbb{R}^n$ is a function $f : \mathbb{R}^n \to \mathbb{R}^n$ for which there exists a number $0 \leq k < 1$ such that

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq k, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n,$$

where $||\cdot||$ is the L^2-norm (Definition A.1). ▲

Definition A.13 (Invariant Set)

We say a set $K \subset \mathbb{R}^n$ is *invariant* with respect to S if there exists a finite set of contraction maps $S = \{S_1, S_2, \ldots, S_N\}$ (Definition A.12) on K such that [Hutchinson, 1981, p.2]

$$K = \bigcup_{i=1}^{N} S_i K. \qquad \blacktriangle$$

Definition A.14 (Lipschitz Constant)

For a function $f : \mathbb{R}^n \to \mathbb{R}^n$, the *Lipschitz constant* is defined as [Hutchinson, 1981, p. 4]

$$\mathrm{Lip}(f) = \sup_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|}{\|\boldsymbol{x} - \boldsymbol{y}\|}.$$

We say that f is Lipschitz continuous if $\mathrm{Lip}(f) < \infty$. ▲

Definition A.15 (Regular Function)

We say that a function $f \in L^2(\mathbb{R})$ is regular if and only if it is continuously differentiable and satisfies [Mallat, 1989, Eq. (15)]

$$\exists C > 0 \forall t \in \mathbb{R} : |f(t)| \leq \frac{C}{1+t^2} \quad \text{and} \quad |f'(t)| \leq \frac{C}{1+t^2}$$

where f' denotes the derivative of f. ▲

Theorem A.16 (Poisson Summation Formula)

If $f : \mathbb{R}^n \to \mathbb{R}$ is a function with Fourier transform $\hat{f}(\cdot)$ for which $|f(t)| \leq C|t|^{-1-\epsilon}$ and $|\hat{f}(\epsilon)| \leq C|\epsilon|^{-1-\epsilon}$ for some $\epsilon > 0$, then [Folland, 2009, Eq. (9.38)]

$$\sum_{k \in \mathbb{Z}} f(k) = \sum_{k \in \mathbb{Z}} \hat{f}(2\pi k). \qquad \blacktriangle$$

Proof

Proof of this theorem can be found in [Folland, 2009, p.337]. ■

Lemma A.17

Let A be a positive trigonometric polynomial consisting only of cosines, i.e. $A(\omega) = \sum_{n=0} a_n \cos(n\omega)$, with $a_n \in \mathbb{R}$ for $0 \leq n \leq N$. Then there exists a trigonometric polynomial B of order N, $B(\omega) = \sum_{n=0}^{N} b_n e^{\mathrm{j}n\omega}$, with $b_n \in \mathbb{R}$ for $0 \leq n \leq N$, such that [Daubechies, 1988, Lemma 4.2]

$$|B(\omega)|^2 = A(\omega) \qquad \blacktriangle$$

Proof

This proof is due to [Daubechies, 1988].

Using that $cos(n\omega) = \frac{1}{2}\left(e^{\mathrm{j}n\omega} - e^{\mathrm{j}n\omega}\right)$, we have

$$\begin{aligned} A(\omega) &= a_0 + \frac{1}{2}\sum_{n=1}^{N} a_n \left(e^{\mathrm{j}n\omega} + e^{\mathrm{j}n\omega}\right) \\ &= e^{-\mathrm{j}N\omega}\left(a_0 e^{\mathrm{j}N\omega} + \frac{1}{2}\sum_{n=1}^{N} a_n \left(e^{\mathrm{j}(n-N)\omega} + e^{-\mathrm{j}(n+N)\omega}\right)\right) \\ &= e^{-\mathrm{j}N\omega}\left(\frac{1}{2}\sum_{n=0}^{N-1} a_{N-n} e^{\mathrm{j}N\omega} + \frac{1}{2}\sum_{n=1}^{N} a_n e^{\mathrm{j}(N+n)\omega}\right). \end{aligned}$$

Let P_A be defined as the polynomial

$$P_A(z) = \frac{1}{2}\sum_{n=0}^{N-1} a_{N-n} z^n + a_0 z^N + \frac{1}{2}\sum_{n=0}^{N} a_n z^{N+n},$$

which has $2N$ zeros. Since $P_A(e^{\mathrm{j}\omega} = e^{\mathrm{j}N\omega}A(\omega)$ and $P_A(e^{-\mathrm{j}\omega} = e^{-\mathrm{j}N\omega}A(\omega)$, we have that $P_A(z) = z^{2N}P_A(z^{-1})$ for $|z| = 1$. Since the two polynomials $P_A(z)$ and $z^{2N}P_A(z^{-1})$ agree on the unit circle, they agree on the entire complex plane and have the same zeros. Thus, if $P_A(z_0) = 0$, then $P_A(z_0^{-1})$ is zero. Additionally, due to the coefficients a_n being real, $(P_A(z))^* = P_A(z^*)$, implying that any complex conjugate of a zero of P_A is also a zero of P_A. Thus, complex zeros of P_A appears in groups of four $(z_0, z_0^*, z_0^{-}1, (z_0^{-1})^*)$, while complex zeros of P_A appears as pairs (z_0, z_0^{-1}). Arranging P_A in terms of its pairs of real zeros $r_k, r_k^{-1} \quad 1 \leq k \leq K$ and its groups of complex zeros $z_j, z_j^*, z_j^{-1}, (z_j^{-1})^*, \quad 1 \leq j \leq J$, we have

$$P_A(e^{\mathrm{j}\omega}) = \frac{1}{2}a_N \left(\prod_{k=1}^{K}(e^{\mathrm{j}\omega} - r_k)(e^{\mathrm{j}\omega} - r_k^{-1})\right)\left(\prod_{j=1}^{J}(e^{\mathrm{j}\omega} - z_j)(e^{\mathrm{j}\omega} - z_j^*)(e^{\mathrm{j}\omega} - z_j^{-1})(e^{\mathrm{j}\omega} - (z_0^{-1})^*)\right).$$

Note that

$$|(e^{\mathrm{j}\omega} - z_0)(e^{\mathrm{j}\omega} - (z_0^{-1})^*)| = |z_0|^{-1}|e^{\mathrm{j}\omega} - z_0|^2.$$

Consequently, leveraging that $A(\omega)$ is a positive trigonometric polynomial, we have

$$\begin{aligned} A(\omega) &= |A(\omega)| \\ &= |P_A(e^{\mathrm{j}\omega})| \\ &= \left(\frac{1}{2}|a_N| \prod_{k=1}^{K}|r_k|^{-1}\prod_{j=1}^{J}|z_j|^{-2}\right)\left|\prod_{k=1}^{K}(e^{\mathrm{j}\omega} - r_k)\prod_{j=1}^{J}(e^{\mathrm{j}\omega} - z_j)(e^{\mathrm{j}\omega} - z_j^*)\right|^2 \\ &= |B(\omega)|^2, \end{aligned}$$

in which $B(\omega)$ is the trigonometric polynomial of order N given as

$$B(\omega) = \left(\frac{1}{2}|a_N| \prod_{k=1}^{K}|r_k|^{-1}\prod_{j=1}^{J}|z_j|^{-2}\right)^{\frac{1}{2}} \prod_{k=1}^{K}(e^{\mathrm{j}\omega} - r_k)\prod_{j=1}^{J}(e^{\mathrm{j}\omega} - z_j)(e^{\mathrm{j}\omega} - z_j^*),$$

with $\mathrm{Re}(z_j)$ denoting the real part of z_j. ■

A.2 Fourier Analysis

Definition A.18 (Continuous Fourier Transform)

Given a real-valued integrable function $f(t)$, the (continuous) Fourier transform of $f(t)$ is [Folland, 2009, p. 213]

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} e^{-\mathrm{j}\omega t} f(t)dt,$$

provided that the integral exists. ▲

Definition A.19 (Discrete Fourier Transform)

Given a sequence of N complex numbers x_n, the discrete Fourier transform (abbr. DFT) of x_n is

$$\hat{x}_k = \sum_{n=0}^{N-1} e^{-\mathrm{j}\frac{2\pi}{N}kn} x_n \qquad \blacktriangle$$

A.3 Probability and Stochastic Processes

Theorem A.20 (Bayes' Theorem)

Consider a random variable with sample space S. Let $B_1, B, 2, \ldots, K$ be a sequence of events such that

$$P(B_k) > 0, \quad k = 1, 2, \ldots, K.$$
$$B_i \cap B_j = \emptyset, \quad i \neq j.$$
$$S = \bigcup_{k=1}^{K} B_k.$$

Then, given an event A with $P(A) > 0$, for any of the events B_j, we have [Olofsson and Andersson, 2012, Prop. 1.11]

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_{k=1}^{K} P(A|B_k)P(B_k)}. \tag{A.1}$$

▲

Proof

By the law of total probability [Olofsson and Andersson, 2012, Th. 1.1], the denominator of (A.1) is $P(A)$. To show that (A.1) holds, we start by rearranging it:

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{P(A)}$$
$$P(B_j|A)P(A) = P(A|B_j)P(B_j),$$

and see that the latter equation is true, since both sides equal $P(A \cap B_j)$. ■

Definition A.21 (Lognormal Distribution)

Let $X \sim \mathcal{N}(\mu, \sigma^2)$. The random variable $Y = e^X$ is said to have a *lognormal distribution* with parameters μ and σ^2 [Olofsson and Andersson, 2012, Def. 2.19]. Its probability density function is given as [Olofsson and Andersson, 2012, Corr. 2.6]

$$f_Y(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{(\log(x)-\mu)^2}{2\sigma^2}}, \quad x > 0.$$

The shorthand used for this distribution is $\log\mathcal{N}(\mu, \sigma^2)$. ▲

Definition A.22 (Gamma Distribution)

If a random variable X has a probability density function given as [Olofsson and Andersson, 2012, Def. 2.20]

$$f_X(x) = e^{-\lambda x} \lambda^\alpha \frac{x^{\alpha-1}}{\Gamma(\alpha)}, \quad x \geq 0,$$

with $\Gamma(\alpha) = \int_0^\infty e^{-t} t^{\alpha-1} dt$, the random variable X is said to have a *gamma distribution* with shape parameters $\alpha > 0$ and $\lambda > 0$. ▲

Definition A.23 (Empirical Distribution Function)

The *empirical distribution function* or *empirical cumulative density function* is used to estimate the cumulative density function of a distribution based on samples from that distribution. Given a set of N random variables $X_1, X_2, \dots, X_N$, the empirical distribution function is defined as [Olofsson and Andersson, 2012, p.369]

$$\hat{F}_N(x) = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}_{\leq x}(X_n),$$

where $\mathbf{1}_{\leq x}(X_n) = 1$ when $X_n \leq x$ and 0 otherwise. In other words, the empirical distribution function of x is the proportion of samples in the sample set less than or equal to x. ▲

Definition A.24 (Kolmogorov-Smirnov Test)

The *Kolmogorov-Smirnov test* is used to assess whether a set of samples is likely to come from a particular distribution. Assuming that we have a set of random samples $X_1, X_2, \dots, X_N$ with cumulative density function $F(x)$, we want to test the hypothesis [Olofsson and Andersson, 2012, p. 369]

$$\begin{aligned} H_0 \quad &: \quad F(x) = F_0(x) \quad \forall x, \\ H_A \quad &: \quad F(x) \neq F_0(x) \quad \text{for some } x, \end{aligned}$$

for a given $F_0(x)$. Letting $\hat{F}_n(x)$ denote the empirical distribution function Definition A.23, the Kolmogorov-Smirnov test statistic is given as

$$D_n = \max_x \|\hat{F}_n(x) - F_0(x)\|.$$ ▲

Theorem A.25 (Jensen's Inequality)

Let X be a random variable and $\phi(\cdot)$ be a convex function. Then [Bishop, 2009, Eq. (1.116)]

$$\phi\left(\mathbb{E}[X]\right) \leq \mathbb{E}\left[\phi(X)\right].$$

If $\phi(\cdot)$ is concave, the direction of the inequality is flipped. The distance between the two sides, $|\mathbb{E}\left[\phi(X)\right] - \phi\left(\mathbb{E}[X]\right)|$, is called the Jensen gap. ▲

Definition A.26 (Wide Sense Stationarity)

A (discrete) stochastic process $(X_n)_{n\in\mathbb{Z}}$ is said to be *wide sense stationary* if it has constant mean and

$$\mathbb{E}\left[X_{n_1} X_{n_2}\right] = f(|n_2 - n_1|),$$

i.e. if the covariance of X_{n_1} and X_{n_2} depends only on the difference between n_1 and n_2. [Kay, 2006, p. 550] ▲

Definition A.27 (Symmetric Alpha-Stable Process)

a (discrete) stochastic process $(X_n)_{n\in\mathbb{Z}}$ is called *symmetric α-stable* with stability parameter $\alpha \in (0,2)$ if, for any finite set of indices $\boldsymbol{n} = [n_0, n_1 \dots, n_{N-1}]$ and constants $\boldsymbol{\theta} = [\theta_0, \theta_1 \dots, \theta_{N-1}] \in \mathbb{R}^N$, we have [Pipiras and Taqqu, 2017, Def. 1.2]

$$\mathbb{E}\left[e^{j\theta \sum_{k=0}^{N-1} \theta_k X_{n_k}}\right] = e^{-\sigma^{\alpha}(\boldsymbol{\theta},\boldsymbol{n})|\theta|^{\alpha}},$$

where $\sigma(\boldsymbol{\theta}, \boldsymbol{n}) > 0$ is a scale parameter. ▲

A.3.1 Markov Chains

Definition A.28 (Discrete-Time Markov Chain)
Let $X_0, X_1, \ldots$ be a sequence of discrete random variables taking values from a set S and let

$$P(X_{n+1} = j | X_0 = i_0, X_0 = i_1 \ldots, X_n = i_n) = P(X_{n+1} = j | X_n = i_n),$$

for all $n \in \mathbb{N}$ and all $j, i_0, i_1, \ldots, i_n \in S$. The sequence $X_0, X_1, \ldots$ is called a *Markov chain* [Olofsson and Andersson, 2012, Def. 8.1]. In other words, a Markov chain is a sequence of values for which future entries, when conditioned on the current entry, are independent of past entries.▲

Definition A.29 (Transition Matrix of Markov Chain)
Let the *state space* S be a discrete set with a cardinality of N. Let $X_0, X_1, \ldots$ be a Markov chain taking values from S and let the transition probabilities $p_{i,j}$ be defined as

$$p_{i,j} = P(X_{n+1} = j | X_N = i), \quad j, i \in S.$$

The *transition matrix* of the Markov chain $X_0, X_1, \ldots$ is defined as the matrix $\boldsymbol{P} \in \mathbb{R}^{N \times N}$, whose (i, j)th entry is $p_{i,j}$ [Olofsson and Andersson, 2012, p. 446]. In theory, N may be infinite, but for this thesis it is sufficient to assume that N is finite. ▲

Definition A.30 (Stationary Distribution of Markov Chain)
Let S be a discrete set with a cardinality of N. Let $X_0, X_1, \ldots$ be a Markov chain with state space S and transition matrix $\boldsymbol{P}$. A probability distribution $\boldsymbol{\pi} \in \mathbb{R}^N$ satisfying

$$\boldsymbol{\pi}^T \boldsymbol{P} = \boldsymbol{\pi}^T$$

is called a *stationary distribution* of the Markov chain [Olofsson and Andersson, 2012, Def. 8.5]. ▲

Definition A.31 (Irreducibility)
Let $X_0, X_1, \ldots$ be a Markov chain with state space S and n-step transition probabilities $p_{i,j}^{(n)} = P(X_{k+n} = j | X_k = i)$. If $p_{i,j} > 0$ for some n, we say that j is *accessible* from i. If i is also accessible from j, we say that i and j *communicate*. If all states in S communicate, the Markov chain is said to be *irreducible*. [Olofsson and Andersson, 2012, Defs. 8.2, 8.3] ▲

Definition A.32 (Recurrent and Transient States)
Let $X_0, X_1, \ldots$ be a Markov chain with state space S. Consider a state $i \in S$ and let τ_i denote the number of steps before the Markov chain first visits state i, i.e.

$$\tau_i = \min\{n \geq 1 : X_n = i\},$$

with $\tau_i = \infty$ if i is never visited. If $P(\tau_i < \infty) = 1$, i is said to be *recurrent*. Otherwise, it is said to be *transient*. Furthermore, if $\mathbb{E}[\tau_i] < \infty$, i is said to be *positive recurrent*, and if $\mathbb{E}[\tau_i] = \infty$, it is said to be *null recurrent*. [Olofsson and Andersson, 2012, Defs. 8.4, 8.6] ▲

Definition A.33 (Periodicity)
Let $X_0, X_1, \dots$ be a Markov chain with n-step transition probabilities $p_{i,j}^{(n)} = P(X_{k+n} = j | X_k = i)$. The *period* of a state i is given as

$$d(i) = \gcd\{n \geq 1 : p_{i,i}^{(n)} > 0\},$$

with gcd denoting the greatest common denominator. If $d(i) > 1$, state i is said to be *periodic*, and if $d(i) = 1$, state i is said to be *aperiodic*. If all states in the state space are aperiodic, the Markov chain is said to be aperiodic. [Olofsson and Andersson, 2012, Def. 8.8] ▲

Theorem A.34 (Convergence to Stationary Distribution)
Consider an irreducible, positive recurrent and aperiodic Markov chain with stationary distribution $\boldsymbol{\pi}$ and transition matrix $\boldsymbol{P}$. Let the n-step transition matrix $\boldsymbol{P}^{(n)}$ denote the nth "power" of $\boldsymbol{P}$, i.e. the matrix product of P with itself $n-1$ times. Then

$$\lim_{n\to\infty} p_{i,j}^{(n)} = \pi_j,$$

for all i, j [Olofsson and Andersson, 2012, Th. 8.1]. ▲

The Wavelet Transformation B

The wavelet transformation is a useful tool in assessing the fractal properties of functions on $\mathbb{R}$. While the next chapter will delve into the meaning of 'fractal properties', fundamental theory about the wavelet transformation will prove useful later when assessing the fractal properties of time series.

B.1 The Wavelet Transform

The Fourier transform (Definition A.18) of a real-valued continuous function of time is a complex-valued continuous function of frequency that expresses the function of time in terms of a set of basis functions on the form $e^{\mathrm{j}\omega t}$, where $t \in \mathbb{R}$ is the time variable and $\omega \in \mathbb{R}$ is the frequency variable. The Fourier transform of a function is purely in the frequency domain and does not provide location information for the occurrences of different frequencies. In some cases, such information could prove valuable. For example, if the function of time represented a sound signal, both frequency and temporal information would be of interest. In these cases, an alternative to the Fourier transform may be needed.

The *wavelet transform* of a function of time is a real or complex-valued function of two variables, $a > 0$ and $b \in \mathbb{R}$, which expresses the function of time in terms of basis functions called *wavelets*, which are on the form $\frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$. The function $\psi(t)$ - referred to as the *mother wavelet* - is an oscillating function that begins and ends with an amplitude of zero. The choice of mother wavelet depends on the application and will be discussed later in this section. The value of a controls the *dilation* of the wavelet. Large values of a yield large-scale wavelets of longer duration and lower frequency. The variable b is called the *translation*. It shifts the wavelet linearly in time, allowing for the wavelet transform to be localized in time. The wavelet transform is localized in both time and frequency as opposed to the Fourier transform, which is only localized in frequency. More accurately, we say that the wavelet transform is localized in time and *scale*. Formally, the wavelet transform is defined as follows:

Definition B.1 (Continuous wavelet transform)

Given a real-valued function $f(t)$ and a mother wavelet $\psi(t)$, the continuous wavelet transform of $f(t)$ is [Simonsen et al., 1997, Eq. (3),(4)]

$$\tilde{f}(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \psi^* \left(\frac{t-b}{a} \right) f(t) dt, \tag{B.1}$$

provided that the integral exists. The parameters $a > 0$ and $b \in \mathbb{R}$ are referred to as the *dilation* and *translation*, respectively. ▲

The choice of mother wavelet $\psi(t)$ in (B.1) depends on the application. Firstly, in order to guarantee that the integral in (B.1) exists, $\psi(t)$ must be square integrable (See the remark following Definition A.2). A square integrable function $\psi(t)$ can be used as a mother wavelet if it decreases rapidly toward 0 for $t \to \pm\infty$ or has compact support (Definition A.7). To ensure this, wavelets are chosen as to satisfy the *admissibility condition.*

Definition B.2 (Admissibility Condition)
Letting $\hat{\psi}(\omega)$ denote the Fourier transform (Definition A.18) of $\psi(t)$, $\psi(t)$ is said to satisfy the admissibility condition if

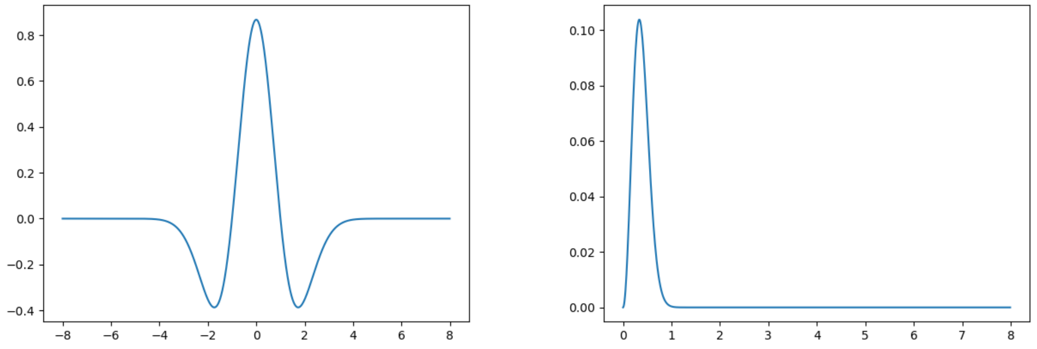
$$\int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega < \infty.$$

holds true [Daubechies, 1988, Eq. (1.3)]. ▲

The admissibility condition Definition B.2 can only hold true if $\hat{\psi}(\omega)$ vanishes at $\omega = 0$, i.e. if $|\hat{\psi}(0)|^2 = 0$. This implies that the function $\psi(t)$ is zero-mean. In other words, [Daubechies, 1988, Eq. (1.4)]

$$\int_{-\infty}^{\infty} \psi(t) dt = 0. \tag{B.2}$$

Hence, the function $\psi(t)$ will have some oscillations.



(a) The Ricker wavelet.

(b) The Fourier transform of the Ricker wavelet.

Figure B.1. The Ricker wavelet - also known as the Mexican hat wavelet - and its Fourier transform.

Depending on the application, it may be necessary to restrict the parameters a, b to a discrete subset of $\mathbb{R}_{\neq 0} \times \mathbb{R}$ (where $\mathbb{R}_{\neq 0} = \mathbb{R} \setminus \{0\}$). This can be achieved by fixing a dilation step $\alpha > 1$ and a translation step $\beta \neq 0$, and considering a discrete set of parameters a and b given as $a_m = \alpha^m$

and $b_{m,n} = n\beta\alpha^m$, respectively, where $m, n \in \mathbb{Z}$ [Daubechies, 1988]. The family of wavelets related to this choice of discretization of a and b are on the form

$$\psi_{m,n}(t) = \frac{1}{\sqrt{\alpha}^m}\psi(\alpha^{-m}t - n\beta), \quad m, n \in \mathbb{Z}. \tag{B.3}$$

This discretization of a and b gives rise to the *discrete wavelet transform.*

Definition B.3 (Discrete Wavelet Transform)
Given a real-valued function $f(t)$, a mother wavelet $\psi(t)$, a dilation step $\alpha > 1$ and a translation step $\beta \neq 0$, the discrete wavelet transform of $f(t)$ is [Daubechies, 1988, Eq. (1.7)]

$$\tilde{f}[m,n] = \frac{1}{\sqrt{\alpha}^m}\int_{-\infty}^{\infty}\psi^*\left(\frac{t}{\alpha^m} - n\beta\right)f(t)dt$$

The value of $\tilde{f}[m,n]$ is referred to as the m,nth *wavelet coefficient.* ▲

In general, the discrete wavelet transform is not guaranteed to have a bounded inverse, i.e. it is not guaranteed that

$$A\int_{-\infty}^{\infty}|f(t)|^2dt < \sum_{m,n\in\mathbb{Z}}|\hat{f}[m,n]|^2 < B\int_{-\infty}^{\infty}|f(t)|^2dt, \tag{B.4}$$

for some $A > 0$ and $B < \infty$. In case (B.4) holds for all square-integrable functions $f(t) \in L^2(\mathbb{R})$, the set $\{\psi_{m,n} : m, n \in \mathbb{Z}\}$ is called a *frame* [Daubechies, 1988]. In this case, $f(t)$ can be reconstructed from its wavelet coefficients as

$$f(t) = \frac{2}{A+B}\sum_{m,n}\psi_{m,n}(t)\tilde{f}[m,n] + R(t),$$

in which $R(t)$ is the error term and satisfies

$$\|R\|_2 \leq O\left(\frac{B}{A} - 1\right)\|f\|_2^2,$$

where $\|\cdot\|_2$ denotes the L^2-norm (Definition A.1).

For values of α and β in Definition B.3 very close to 1 and 0, respectively, the discrete wavelet transform is similar to the continuous case defined in Definition B.1. In this case, it is possible that the functions $\psi_{m,n}(t)$ are linearly dependent and that, consequently, there is a significant amount of redundancy in the set of functions $\{\psi_{m,n} : m, n \in \mathbb{Z}\}$ [Daubechies, 1988]. Depending on the application, redundancy may or may not be desirable. For the purposes of this project, we focus on applications in which redundancy is undesirable. In order to reduce the amount of redundancy in the discrete wavelet transform, the mother wavelet $\psi(t)$ and the parameters α and β can be chosen such that the set $\{\psi_{m,n} : m, n \in \mathbb{Z}\}$ constitutes an orthonormal basis for $L^2(\mathbb{R})$ (Definition A.10). A simple example of an orthonormal basis of wavelets is the *Haar basis* in Example B.1.

Example B.1 (Haar basis)

One example of an orthonormal basis of wavelets is the Haar basis, in which the mother wavelet $\psi_H(t)$ is given as [Folland, 2009, p. 202]

$$\psi_H(t) = \begin{cases} 1, & 0 \le t < \frac{1}{2}, \\ -1, & \frac{1}{2} \le t < 1, \\ 0, & \text{otherwise.} \end{cases}$$

We immediately see that $\psi_H(t)$ has compact support (specifically, it is supported in $[0,1]$) and satisfies the admissibility condition Definition B.2. Choosing $\alpha = 2$ and $\beta = 1$ yields a discretized set of wavelet functions on the form of (B.3) given as

$$\psi_{H;m,n}(t) = \frac{1}{\sqrt{2}^m}\psi_H(2^{-m}x - n), \quad m,n \in \mathbb{Z}.$$

These wavelet functions constitute an orthonormal basis for $L^2(\mathbb{R})$. To show this, we must show that the L^2-norm of $\psi_{H;m,n}$ is 1 for any m,n, that the inner product between any two functions $\psi_{H;m_1,n_1}$ and $\psi_{H;m_2,n_2}$ is zero whenever $m_1 \neq m_2$ or $n_1 \neq n_2$ and that $\{\psi_{H;m,n}, m,n \in \mathbb{Z}\}$ span $L^2(\mathbb{R})$. We start by showing the former:

$$\int_{-\infty}^{\infty} \left| \frac{1}{\sqrt{2}^m}\psi_H(2^{-m}t - n)\right|^2 dt = \int_{2^m n}^{2^m(n+1)} \frac{1}{2^m} dt = 1$$

where we use that $|\psi_H(t)|^2 = 1$ for $t \in [0,1]$ and 0 otherwise. Having shown that the functions $\psi_{H;m,n}$ are of unit length, we will now assess whether they are orthogonal and consider the inner product of two functions $\psi_{H;m_1,n_1}(t)$ and $\psi_{H;m_2,n_2}$, which we choose to denote by $\psi_1(t)$ and $\psi_2(t)$, respectively, for ease of notation.

$$\int_{-\infty}^{\infty} \psi_1(t)\psi_2^*(t)dt = \int_{\text{supp}(\psi_1)\cap\text{supp}(\psi_2)} \psi_1(t)\psi_2^*(t)dt, \tag{B.5}$$

since the product of the two functions is only non-zero at the intersection of their support, the bounds of the integral are changed to reflect this. To proceed, we consider the intersection for the two different cases: $m_1 = m_2, n_1 < n_2$ and $m_1 < m_2$.

The first case ($m_1 = m_2, n_1 < n_2$) is trivial, since $\text{supp}(\psi_1) = [2^m n_1, 2^m(n_1+1)]$ and $\text{supp}(\psi_2) = [2^m n_2, 2^m(n_2+1)]$ are disjoint, i.e. $\text{supp}(\psi_1) \cap \text{supp}(\psi_2) = \emptyset$.

For the second case ($m_1 < m_2$), the functions $\psi_1(t)$ and $\psi_2(t)$ are supported on $[2^{m_1}n_1, 2^{m_1}(n_1+1)]$ and $[2^{m_2}n_2, 2^{m_2}(n_2+1)]$, respectively. The two supports are joint only when $2^{m_2-m_1}n_2 - 1 < n_1 < 2^{m_2-m_1}(n+1)$. In this case, we have that $\text{supp}(\psi_1) \cap \text{supp}(\psi_2) = \text{supp}(\psi_1)$. Since the bounds of $\text{supp}(\psi_1)$ are located at 2^{m_1} scaled by an integer n_1 or $n_1 + 1$ and since $2^{m_1} < 2^{m_2}$, $\text{supp}(\psi_1)$ is contained entirely within one half of $\text{supp}(\psi_2)$. Thus, $\psi_2(t)$ is constant in the integral in (B.5). Since ψ_1 integrates to zero over its support, the integral (B.5) is zero.

Finally, to show that $\{\psi_{H;m,n}, m,n \in \mathbb{Z}\}$ is a basis of $L^2(\mathbb{R})$, we note that since the functions $\psi_{H;m,n}$ are dyadic step functions, the space of linear combinations of $\psi_{H;m,n}$ with $m < M$ equals the space of functions on $\mathbb{R}$ that are constant on each interval $[2^{-M}k, 2^{-M}(k+1)], 0 \le k <$

2^M [Folland, 2009, p.200]. Thus, the set of all finite linear combinations of the Haar wavelets is the set of all step functions on $\mathbb{R}$ with discontinuities at $2^{-m}k, m \in \mathbb{Z}, k \geq 0$. This space is dense (Definition A.11) in $L^2(\mathbb{R})$ and thus, the set $\{\psi_{H;m,n}, m, n \in \mathbb{Z}\}$ is a basis of $L^2(\mathbb{R})$ [Folland, 2009, p.200]. ▲

Orthonormal wavelet bases for function spaces are traditionally used as an intermediate step in obtaining a discrete wavelet transform (Definition B.3) that is essentially a discrete algorithm. Daubechies wavelets, which will be introduced in Appendix B.2, circumvent this intermediate step by imposing a set of necessary and sufficient conditions on the discrete side of the transformation to obtain an algorithm that is fully discrete. The Daubechies discrete wavelet transform is applied as a discrete scheme using filters with a finite number of taps. The Daubechies wavelets are based on a *multiresolution analysis* framework, which will be the topic of the next subsection.

B.1.1 Construction of Orthonormal Wavelet Basis using Multiresolution Analysis

Multiresolution analysis is a valuable tool in the construction of orthonormal wavelet bases and the design of discrete wavelet transforms. We will delve into why this is the case later in this subsection. First, we introduce the *multiresolution approximation* of $L^2(\mathbb{R})$ as introduced in [Mallat, 1989].

Definition B.4 (Multiresolution Approximation)
A multiresolution approximation of $L^2(\mathbb{R})$ is a sequence $\{V_m : m \in \mathbb{Z}\}$ of closed subspaces of $L^2(\mathbb{R})$ such that the following hold [Mallat, 1989]:

1. $V_m \subset V_{m+1}, \quad \forall m \in \mathbb{Z}$.
2. *Completeness*: $\bigcap_{m\in\mathbb{Z}} V_m = \{0\}$ and $\bigcup_{m\in\mathbb{Z}} V_m = L^2(\mathbb{R})$ (The union of the subsets V_m is dense (Definition A.11) in $L^2(\mathbb{R})$).
3. *Self-similarity in scale*: $f(t) \in V_m \iff f(2t) \in V_{m+1}, \forall m \in \mathbb{Z}$.
4. *Self-similarity in time*: $f(t) \in V_m \implies f(t - 2^{-m}k) \in V_m, \forall k \in \mathbb{Z}$.
5. There exists an isomorphism $I : V_0 \to l^2(\mathbb{Z})$ (see Definition A.4), which commutes with the action of $\mathbb{Z}$. ▲

By "commutes with the action of $\mathbb{Z}$" in property (5), we mean that the isomorphism commutes with translation of integers, i.e. for an isomorphism $I : V_0 \to l^2(\mathbb{Z})$, a function $f(t) \in V_0$ and an l^2-sequence $I\big(f(t)\big) = x[n]$, we have [Mallat, 1989, p.71]

$$I(f(x-k)) = x[n-k], \quad k \in \mathbb{Z}.$$

Before delving into how orthonormal wavelet bases may be constructed using multiresolution analysis, a brief summary of some properties of the multiresolution approximation will be necessary. Given a regular (Definition A.15) function $\phi(t)$ for which the family of functions $\sqrt{2^m}\phi(2^m t-n),\ k\in\mathbb{Z}$ is an orthonormal family for all $m\in\mathbb{Z}$, we let V_m denote the vector space generated by this family of functions, such that the set of vector spaces $V_m,\ m\in\mathbb{Z}$ is a multiresolution approximation of $L^2(\mathbb{R})$. Property (2) of Definition B.4 implies

$$\frac{1}{2}\phi\left(\frac{t}{2}\right)\in V_{-1}\subset V_0,$$

and hence, the function can be decomposed in terms of the basis functions $\phi(t-n)$ of V_0 [Mallat, 1989, Eq. (16)]:

$$\frac{1}{2}\phi(\frac{t}{2})=\sum_{n\in\mathbb{Z}}h[n]\phi(t+n),\quad \text{where } h[n]=\frac{1}{2}\int_{-\infty}^{\infty}\phi(\frac{t}{2})\phi^*(t+n)dt.$$

Since $\phi(t)$ is regular, we have asymptotic decay of $h[n]$ with $|h[n]|=O(1+k^2)^{-1}$ [Mallat, 1989]. Taking the Fourier transform (Definition A.18) yields [Mallat, 1989, Eq. (17)]

$$\hat{\phi}(2\omega)=h(\omega)\hat{\phi}(\omega),\quad \text{where } h(\omega)=\sum_{n\in\mathbb{Z}}h[n]e^{-\mathrm{j}n\omega}, \tag{B.6}$$

where $h(\omega)$ satisfies $|h(\omega)|^2+|H(\omega+\pi)|^2$ and $|H(0)|=1$. Proof of this can be found in [Mallat, 1989, Th. 1].

Consider again the multiresolution approximation Definition B.4. The larger m is, the more accurately the projection of a function $f(t)\in L^2(\mathbb{R})$ onto V_m approximates the original function. The amount of additional information available at resolution 2^{m+1} compared to resolution 2^m is the orthogonal complement of V_m in V_{m+1}, which we choose to denote by W_m. In other words, W_m is defined by the following relation for $m\in\mathbb{Z}$:

$$V_{m+1}=V_m\oplus W_m,\quad V_m\perp W_m. \tag{B.7}$$

We aim to show that there exists a function $\psi(t)$ such that the family of functions $\sqrt{2^m}\psi(2^m t-n),\ k\in\mathbb{Z}$ is an orthonormal basis of W_m and, by extension, that the family of functions $\sqrt{2^m}\psi(2^m t-n),\ m,n\in\mathbb{Z}$ is an orthonormal basis of $L^2(\mathbb{R})$.

From (3) of Definition B.4 and (B.7), it follows that it must hold true for the function $\psi(t)$ that $\psi(\frac{t}{2})\in W_{-1}\subset V_0$. Since a scaling in time corresponds to an inversely proportional scaling in frequency, the Fourier transform (Definition A.18) of $\psi(t)$ can be written as [Mallat, 1989, Eq. 63]

$$\hat{\psi}(2\omega)=g(\omega)\hat{\phi}(\omega), \tag{B.8}$$

where $\phi(t)\in V_0$ and $g(\omega)$ is a 2π-periodic $L^2([0,2\pi])$-function.

Recalling (B.7), we have that the Fourier transform of any $f(t)\in V_0$ can be decomposed in terms of the spaces V_{-1} and W_{-1}, i.e. [Mallat, 1989, Eq. (64)]

$$\hat{f}(\omega)=a(\omega)\hat{\phi}(\omega)=b(\omega)\hat{\phi}(2\omega)+c(\omega)\hat{\psi}(2\omega),$$

where $a(\omega)$ is a 2π-periodic $L^2([0,2\pi])$-function and $b(\omega)$ and $c(\omega)$ are π-periodic $L^2([0,\pi])$-functions. By inserting (B.6) and (B.8) into the equation above, we see that

$$a(\omega) = b(\omega)h(\omega) + c(\omega)g(\omega). \tag{B.9}$$

The orthogonality condition for this decomposition can be stated as

$$\int_0^{2\pi} |a(\omega)|^2 d\omega = \int_0^{\pi} |b(\omega)|^2 d\omega + \int_0^{\pi} |c(\omega)|^2 d\omega,$$

which is satisfied if and only if [Mallat, 1989, Eq. (66)]

$$\begin{aligned} &|h(\omega)|^2 + |g(\omega)|^2 = 1, \\ &h(\omega)g^*(\omega) + h(\omega+\pi)g^*(\omega+\pi) = 0. \end{aligned} \tag{B.10}$$

Under these conditions on $g(\omega)$, $\psi(t)$ can be constructed from (B.8). One possible choice for G given in [Mallat, 1989, Eq. (69)] is

$$g(\omega) = e^{-\mathrm{j}\omega} h^*(\omega+\pi).$$

All that remains to be shown now is that the family of wavelets $\sqrt{2^m}\psi(2^m t - n),\ \ m,n \in \mathbb{Z}$ spans $L^2(\mathbb{R})$. Recall that $\bigcup_{m\in\mathbb{Z}} V_m$ is dense in $L^2(\mathbb{R})$ due to property (2) of Definition B.4. Since the vector space V_m can be obtained as a direct sum of all vector spaces W_j with $j < m$, $\bigoplus_{m\in\mathbb{Z}} W_m$ is also dense in $L^2(\mathbb{R})$. Thus, the family of functions $\sqrt{2^m}\psi(2^m t - n),\ \ m,n \in \mathbb{Z}$ is an orthonormal basis of $L^2(\mathbb{R})$ [Mallat, 1989].

B.2 Daubechies Wavelets

The Daubechies wavelets are a family of wavelets used to define a discrete wavelet transform. They skip the intermediate step of constructing an orthonormal basis of continuous basis functions. Instead, the Daubechies wavelet transform is a completely discrete algorithm, essentially operating as a series of filters. Because of this, the Daubechies wavelet functions cannot be written in closed form. The aim of this subsection is to summarize the main points of Daubechies [1988] regarding the discrete wavelet transform with Daubechies wavelets.

The Daubechies wavelets were motivated by a desire to decompose a square-summable (Definition A.4) sequence $c_0[n] \in l^2(\mathbb{R})$ into multiple levels corresponding to various frequency bands.

This can be done by use of multiresolution analysis, so assume that spaces V_m have been chosen as to that satisfy the properties of Definition B.4 and that a function $\phi(t) \in \mathbb{V}_m$ for which the family of functions $\phi_{m,n}(t) = \sqrt{2^m}\phi(2^m t - n),\ \ k \in \mathbb{Z}$ is an orthonormal family whose closure spans V_m for all $m \in \mathbb{Z}$. We let $\{\psi_{m,n}; m,n \in \mathbb{Z}\}$ denote the associated orthonormal wavelet basis. Additionally, we assume that both ϕ and ψ are real.

We start by constructing a function $f(t)$ given as

$$f(t) = \sum_{n\in\mathbb{Z}} c_0[n]\phi(t-n), \tag{B.11}$$

which is an element of V_0 [Daubechies, 1988]. Recall that since $V_0 = V_{-1} \oplus W_{-1}$, we can decompose f into its components in V_{-1} and W_{-1} as

$$f = P_1 f + Q_1 f, \tag{B.12}$$

where $P_1 f$ and $Q_1 f$ denote the projections of f onto V_{-1} and W_{-1}, respectively. The first term of (B.12) can be written in terms of the orthonormal basis $\phi_{-1,n}$ as

$$P_1 f(t) = \sum_{k\in\mathbb{Z}} c_1[k]\phi_{-1,k}(t). \tag{B.13}$$

Since $P_1 f$ is a version of f at a lower resolution, the sequence $c_1[k]$ represents a smoothed version of $c_0[n]$.

Note that since the $\phi_{-1,k}, \quad k \in \mathbb{Z}$ form an orthonormal basis, the inner product between the projection of f onto V_{-1} (B.13) and a basis function $\phi_{-1,k}$ can be used to compute the coefficient $c_1[k]$ associated with $\phi_{-1,k}$. Specifically, we have that [Daubechies, 1988, p. 935]

$$c_1[k] = \langle \phi_{-1,k}, P_1 f\rangle = \langle \phi_{-1,k}, f\rangle = \sum_{n\in\mathbb{Z}} c_0[n]\langle \phi_{-1,k}, \phi_{0,n}\rangle, \quad k \in \mathbb{Z}, \tag{B.14}$$

where the second equation follows from (B.12) and the fact that $\phi_{-1,k} \in V_{-1}$ is orthogonal to $Q_1 f \in W_{-1}$. The third equation follows from using the definition of f in (B.11) to express the inner product as a linear combination of smaller inner products.

By defining

$$h(n) = \frac{1}{\sqrt{2}} \int_{-\infty}^{\infty} \phi\left(\frac{t}{2}\right)\phi(t-n)dt, \tag{B.15}$$

we can restate (B.14) as [Daubechies, 1988, Eq. (2.47)]

$$\begin{aligned} c_1[k] &= \sum_{n\in\mathbb{Z}} c_0 \langle \phi_{-1,k}, \phi_{0,k}\rangle \\ &= \sum_{n\in\mathbb{Z}} c_0 \int_{-\infty}^{\infty} \phi_{-1,k}(t)\phi_{0,n}(t)dt \\ &= \sum_{n\in\mathbb{Z}} c_0 \int_{-\infty}^{\infty} \frac{1}{\sqrt{2}}\phi(\frac{t}{2}-k)\phi(t-n)dt \\ &\overset{(a)}{=} \sum_{n\in\mathbb{Z}} c_0 \frac{1}{\sqrt{2}} \int_{-\infty}^{\infty} \phi(\frac{t'}{2})\phi(t'-(n-2k))dt \\ &= \sum_{n\in\mathbb{Z}} h(n-2k)c_0[n], \end{aligned} \tag{B.16}$$

where (a) follows from a change of variables $t' = t - 2k$.

The second term of (B.12) can - in a manner similar to the first term - be written in terms of the orthonormal basis $\psi_{-1,n}$ as

$$Q_1 f(t) = \sum_{k\in\mathbb{Z}} d_1[k]\psi_{-1,k}(t). \tag{B.17}$$

Since the projection of f onto W_{-1} can be considered as the information 'lost' by projecting f onto V_{-1} from V_0, the sequence $d_1[k]$ represents the difference in information between $c_0[n]$ and $c_1[k]$. By a process similar to that leading to (B.16), we see that

$$d_1[k] = \sum_{n\in\mathbb{Z}} g(n-2k)c_0[n], \tag{B.18}$$

where

$$g(n) = \frac{1}{\sqrt{2}}\int_{-\infty}^{\infty} \psi\left(\frac{t}{2}\right)\phi(t-n)\,dt. \tag{B.19}$$

This procedure of obtaining $c_1[k]$ and $d_1[k]$ from $c_0[n]$ can be iterated upon. Since $V_{-1} = V_{-2} \oplus W_{-2}$, $P_1 f$ can be further decomposed into its components in V_{-2} and W_{-2}. These components, $P_2 f$ and $Q_2 f$, can in turn be written on forms similar to (B.13) and (B.17). By an approach similar to those previously described in this section, it follows that [Daubechies, 1988, p. 937]

$$c_2[k] = \sum_{n\in\mathbb{Z}} h(n-2k)c_1[n], d_2[k] = \sum_{n\in\mathbb{Z}} g(n-2k)c_1[n],$$

where $h(\cdot)$ and $g(\cdot)$ are as defined in (B.15) and (B.19), respectively.

The procedure may be generalized as follows: Given an initial sequence $c_0[n]$, the decomposition can be performed by iterating

$$c_j[k] = \sum_{n\in\mathbb{Z}} h(n-2k)c_{j-1}[n], \tag{B.20}$$

$$d_j[k] = \sum_{n\in\mathbb{Z}} g(n-2k)c_{j-1}[n], \tag{B.21}$$

starting from $j = 1$. In theory, (B.20) and (B.21) can be iterated infinitely. In practice, however, we are limited by the number of available samples.

Assume that we have decomposed a sequence $c_0[k]$ into a set of 'difference' sequences $d_1[k], d_2[k], \ldots, d_L[k]$ obtained from repeated iteration of (B.20), and a single smoothed sequence $c_L[k]$ obtained from the Lth iteration of (B.20). To reconstruct the original sequence $c_0[k]$, we recall that since $P_{j-1}f = \sum_{k\in\mathbb{Z}} c_j[k]\phi_{j,k} + \sum_{k\in\mathbb{Z}} d_j[k]\psi_{j,k}$, we have (from (B.14), (B.13) and (B.17)) that [Daubechies, 1988, p.938]

$$c_{j-1}[n] = \langle \phi_{-(j-1),n}, P_{j-1}f\rangle = \sum_{k\in\mathbb{Z}} h(n-2k)c_j[k] + \sum_{k\in\mathbb{Z}} g(n-2k)d_j[k]. \tag{B.22}$$

The original sequence $c_0[n]$ can be obtained by repeated iteration of (B.22) starting with $j = L$ and decreasing it at each step.

The sequence $c_0[n]$ can be decomposed according to (B.20) and (B.21), which only requires knowledge of the values of the functions h and g at integer intervals. Thus, this decomposition algorithm lends itself well to being completely discretized. By defining the digital filter taps $h_n = h(n), \quad n \in \mathbb{Z}$ and $g_n = g(n), \quad n \in \mathbb{Z}$, a wavelet decomposition algorithm that is entirely discrete can be obtained by imposing certain conditions on the filter taps h_{n-2k} and g_{n-2k}.

This will be the main topic of the following subsection. To round of this section, we denote the Daubechies wavelet coefficients as

$$\tilde{f}[m,n] = d_m[n], \tag{B.23}$$

to match the notation of Definition B.3. From (B.20) and (B.21) and the discussion following them, it is clear that the wavelet coefficients (B.23) are only defined for $0 \leq m \leq L$, where L is the number of iterations of (B.20) and (B.21) performed. Additionally, in practical applications, the wavelet coefficients are non-zero for only a finite number of values of n.

B.2.1 Filter Coefficients of the Daubechies Wavelet Transform

This subsection will provide a brief summary of the main results of [Daubechies, 1988]. The conditions that must be imposed on the sequences h_n and g_n to obtain a completely discrete wavelet algorithm are stated in the following theorem. This theorem is stated without proof. For proof, see [Daubechies, 1988].

Theorem B.5

Let h_n be a sequence such that [Daubechies, 1988, Th. 3.6]

1. $\sum\limits_{n\in\mathbb{Z}} |h_n||n|^{\epsilon} < \infty$ for some $\epsilon > 0$,

2. $\sum\limits_{n\in\mathbb{Z}} h_{n-2k}h_{n-2l} = \begin{cases} 1, & k = l \\ 0, & k \neq l \end{cases}$

3. $\sum\limits_{n\in\mathbb{Z}} h_n = \sqrt{2}$.

Suppose also that $m_0(\omega) = \frac{1}{\sqrt{2}} \sum_{n\in\mathbb{Z}} h_n e^{\mathrm{j}n\omega}$ can be written as

$$m_0(\omega) = \left(\frac{1}{2}\left(1 + e^{\mathrm{j}\omega}\right)\right)^N \left(\sum_{n\in\mathbb{N}} f(n)e^{\mathrm{j}n\omega}\right), \tag{B.24}$$

where

4. $\sum\limits_{n\in\mathbb{Z}} |f(n)||n|^{\epsilon} < \infty$ for some $\epsilon > 0$,

5. $\sup\limits_{\omega} \left|\sum\limits_{n\in\mathbb{Z}} f(n)e^{\mathrm{j}n\omega}\right| < 2^{N-1}$

Define

$$g_n = (-1)^n h_{-n+1}, \tag{B.25}$$

$$\hat{\phi}(\omega) = \frac{1}{\sqrt{2\pi}} \prod_{i=1}^{\infty} m_0(2^{-i}\omega), \tag{B.26}$$

$$\psi(t) = \sqrt{2} \sum_{n\in\mathbb{Z}} g_n \phi(2t - n), \tag{B.27}$$

where $\hat{\phi}(\cdot)$ denotes the Fourier transform (Definition A.18) of ϕ. Then, the $\phi_{m,n}(t) = 2^{-\frac{m}{2}}\phi(2^{-m}x - n), \quad m,n \in \mathbb{Z}$ define a multiresolution analysis (Definition B.4). The functions $\psi_{m,n}(t) = 2^{-\frac{m}{2}}\psi(2^{-m}x - n), \quad m,n \in \mathbb{Z}$ are the associated wavelet basis. ▲

Condition 1 of Theorem B.5 ensures pointwise convergence of (B.26) for all $\omega \in \mathbb{R}$ [Daubechies, 1988, Lemma 3.1]. Condition 2 imposes orthogonality on the sequence h_n (and g_n, through its definition in (B.25)) [Daubechies, 1988]. Condition 3 ensures that h_n acts as an averaging operator when applied as in (B.20) [Daubechies, 1988]. Conditions 4 and 5 ensure that the function ϕ is continuous [Daubechies, 1988, Lemma 3.2, Remark 1].

Note that using Theorem B.5 as a recipe for constructing orthonormal wavelet bases does not guarantee that the wavelets have compact support. We can ensure that the wavelets have compact support by having the sequence h_n take nonzero values only within a finite interval, while still satisfying the conditions of Theorem B.5 [Daubechies, 1988, Eq. (4.1)-(4.3)].

Assuming that the sequence $h_n, \quad n \in \mathbb{Z}$ has a finite set of non-zero entries, the function $m_0(\omega) = \frac{1}{\sqrt{2}}\sum_{n\in\mathbb{Z}} h_n e^{jn\omega}$ will contain a finite amount of non-zero terms in the sum and (B.24) can be rewritten as [Daubechies, 1988, Eq. (4.8)]

$$m_0(\omega) = \left(\frac{1}{2}\left(1 + e^{\mathrm{j}\omega}\right)\right)^N Q\left(e^{\mathrm{j}}\right), \tag{B.28}$$

where $Q(\cdot)$ is a polynomial whose coefficients are real. Taking the power of (B.28) yields

$$|m_0(\omega)|^2 = \left|\left(\frac{1}{2}\left(1 + e^{\mathrm{j}\omega}\right)\right)^N Q\left(e^{\mathrm{j}}\right)\right|^2 = \left(\cos^2\left(\frac{\omega}{2}\right)\right)\left|Q\left(e^{\mathrm{j}}\right)\right|^2. \tag{B.29}$$

Condition 2 of Theorem B.5 can also be stated as [Daubechies, 1988, Eq. (4.7)]

$$|m_0(\omega)|^2 = |m_0(\omega + \pi)|^2 = 1. \tag{B.30}$$

Using (B.29) and by introducing $z = \cos^2\left(\frac{\omega}{2}\right)$ and $P\left(\sin^2\left(\frac{\omega}{2}\right)\right) = P(1-z) = |Q(e^{\mathrm{j}\omega})|^2$, this becomes

$$z^N P(1-z) + (1-z)^N P(z), \quad z \in [0,1], \tag{B.31}$$

with $P(z) \geq 0, \quad z \in [0,1]$. It follows that any function m_0 that satisfies (B.30) corresponds to a polynomial P that solves (B.31). Conversely, it has been shown that every polynomial P that satisfies (B.31) corresponds to solutions of (B.30) with a real-valued $h_n, \quad n \in \mathbb{Z}$ [Daubechies, 1988, Lemma 4.2].

One possible solution to (B.31) is a polynomial on the form [Daubechies, 1988, Eq. (4.13)]

$$P_N(z) = \sum_{i=0}^{N_1} \binom{N-1+j}{j} z^j, \quad z \in [0,1]. \tag{B.32}$$

Furthermore, it can be shown that any polynomial is on the form [Daubechies, 1988, Eq. (4.15)]

$$P(z) = P_N(y) + z^N R(z), \quad z \in [0,1], \tag{B.33}$$

with R being a polynomial that satisfies [Daubechies, 1988, p. 976]

$$R(1-z)+R(z)=0, \quad z\in[0,1].$$

This polynomial is antisymmetric around $\frac{1}{2}$ and by introducing

$$S\left(\frac{1}{2}-z\right)=R(z), \quad z\in[0,1]$$

(B.33) can be rewritten in terms of the odd polynomial S as

$$P(z)=P_N(z)+z^N S\left(\frac{1}{2}-z\right), \quad z\in[0,1]. \tag{B.34}$$

Using the equation above, m_0 can be constructed by choosing a positive integer N and choosing an odd polynomial S that guarantees that (B.34) satisfies that $P(z)\geq 0, \quad z\in[0,1]$ as well as condition 5 of Theorem B.5.

For the Daubechies family of wavelets, the polynomial Q is chosen to be of minimal order, such that for any choice of integer $N\geq 1$, we have $S=0$ and $|Q(e^{\mathrm{j}\omega})|^2=P_N(1-z)$ (with $z=\cos^2\left(\frac{\omega}{2}\right)$). From (B.32) it is clear that $P(z)\geq 0, \quad z\in[0,1]$ is satisfied. It can also been shown that it satisfies condition 5 of Theorem B.5 [Daubechies, 1988, Eq. (4.23)]. With $|Q(e^{\mathrm{j}\omega})|^2$ known, $Q((e^{\mathrm{j}\omega})$ can be constructed (see either Lemma A.17 or [Daubechies, 1988, Lemma 4.2]). From Q, m_0 can be constructed according to (B.24). From m_0, the h_n are determined as to satisfy $m_0(\omega)=\frac{1}{\sqrt{2}}\sum_{n=0}^{2N-1}h_n e^{\mathrm{j}n\omega}$. For larger N, the h_n are determined numerically.

A beneficial feature of the Daubechies wavelet functions is that they have a high number of *vanishing moments*, which are defined as follows:

Definition B.6 (Vanishing moment)
We say a wavelet function $\psi(t)$ has k *vanishing moments* if it satisfies [Daubechies, 1988]

$$\int_{-\infty}^{\infty} t^j\psi(t)dt=0, \quad j=0,1,\ldots,k-1 \qquad \blacktriangle$$

By using a wavelet function with k vanishing moments, the wavelet coefficients will be zero for polynomials of degree $\leq k-1$. Wavelet functions with a higher number of vanishing moments generally yield a more sparse set of wavelet coefficients. Note that due to the zero-mean property (B.2) of the mother wavelet, any wavelet function has at least one vanishing moment.

Deep Learning C

The aim of this chapter is to introduce the general theory behind neural networks. First, the most basic family of neural network - *feed-forward networks* - will be introduced to allow for a discussion of the fundamentals of neural networks. In the main text of this thesis, another type of network called *convolutional neural networks* were used - these will be introduced in Appendix C.2.

C.1 Feed-Forward Networks

The aim of this section is to introduce fundamental theory of artificial neural networks by using the most straightforward class of neural networks - known as *feed-forward* networks - as a learning example. The class of *convolutional* neural networks (abbr. CNN) will be introduced later in Appendix C.2. This section is based on [Bishop, 2009, Sec. 5.1].

Artificial neural networks are a family of machine learning methods inspired by the information processing systems of biological neural networks. They are commonly used for regression or classification problems, in which the aim of regression is to estimate the value of one or more continuous variables given a vector $\boldsymbol{x} \in \mathbb{R}^N$ of input variables, and the aim of classification is to assign an input vector $\boldsymbol{x}$ to one of a discrete set of classes, which are commonly assumed to be disjoint. A common model used for both regression and classification problems is one based on linear combinations of nonlinear basis functions $h_j(\boldsymbol{x})$, $j = 1, 2, \ldots, M$ given as [Bishop, 2009, Eq. (5.1)]

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M})\, w_j h_j(\boldsymbol{x})\right), \tag{C.1}$$

where the entries of $\boldsymbol{w} = [w_1, w_2, \ldots, w_M]^\intercal$ are called *weights* and the function $f(\cdot)$ is called an *activation function*. Generally, the activation function is a nonlinear differentiable function; a notable exception to this is for regression problems in which the range of a variables to be estimated spans the entirety of $\mathbb{R}$, in which case the identity is used.

The feed-forward network model can be considered a natural extension of (C.1); by making the basis functions $h(\boldsymbol{x})$ themselves dependent on adjustable parameters, which are fitted along-side the weights $w_i, i = 1, 2, \ldots, M$ during training. This is achieved by using basis functions that follow the same form as (C.1) - nonlinear functions of linear combinations of the inputs $\boldsymbol{x}$.

The basic feed-forward neural network model can be described as a series of linear combinations followed by functional transformations. The first step in the neural network is to construct M linear combinations of the input variables $\boldsymbol{x}$ [Bishop, 2009, Eq. (5.2)]:

$$a_j^{(1)} = \sum_{i=1}^{N} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}, \quad j = 1, 2, \ldots, M, \tag{C.2}$$

where we use the superscript (1) for the parameters to indicate that they are in the first *layer* of the network. We call the dependent variables a_j *activations*. By applying an activation function to the activations, we obtain the *hidden units* given as [Bishop, 2009, Eq. (5.3)]

$$z_j^{(1)} = h_1(a_j^{(1)}), \quad j = 1, 2, \ldots, M. \tag{C.3}$$

The choice of activation function depends on the application and will be discussed in Appendix C.1.1. For now, we'll simply note that the activation functions must be nonlinear and differentiable. We require nonlinearity since any linear function of a linear combination is just a linear combination. The reason behind the requirement of differentiability will become apparent in Appendix C.1.3.

By a linear combination of the hidden units, the output unit activations are obtained [Bishop, 2009, Eq. (5.4)]:

$$a_k^{(2)} = \sum_{j=1}^{M} w_{k,j}^{(2)} z_j^{(1)} + w_{k,0}^{(2)}, \quad k = 1, 2, \ldots, K. \tag{C.4}$$

To obtain a model similar to (C.1), these activations can be transformed by an appropriate activation function to produce outputs $y_k, k = 1, 2, \ldots, K$:

$$y_k = f(a_k^{(2)}), \quad k = 1, 2, \ldots, K. \tag{C.5}$$

Alternatively, another layer of hidden units can be added to the model before producing the outputs by making use of (C.3) and (C.3), substituting layers 1 and 2 for layers 2 and 3, respectively. The outputs and hidden units are collectively referred to as the *neurons* of the neural network.

Combining (C.2), (C.3) and (C.4) and applying an output activation function $f(\cdot)$, we obtain the overall network function as

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{k,j}^{(2)} h_1 \left(\sum_{i=1}^{N} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right), \quad k = 1, 2, \ldots, K, \tag{C.6}$$

in which $\boldsymbol{w}$ contains all the weights $\{w_m, n^{(i)}\}$. This model is simply a nonlinear function of the inputs $\boldsymbol{x}$ parameterized by the weights $\boldsymbol{w}$, which may be adjusted during the fitting of the model. The adjustment of the weights $\boldsymbol{w}$ will be discussed in Appendix C.1.3.

The neural network model (C.6) can be simplified by defining an additional input parameter $x_0 = 1$ (referred to as the bias parameter), allowing for (C.2) to be written as

$$a_j^{(1)} = \sum_{i=0}^{N} w_{j,i}^{(1)} x_i, \quad j = 1,2,\dots,M. \tag{C.7}$$

Similarly, we may define an additional hidden unit $z_0^{(1)} = 1$, so that (C.4) can be written as

$$a_k^{(2)} = \sum_{j=0}^{M} w_{k,j}^{(1)} z_j, \quad k = 1,2,\dots,K. \tag{C.8}$$

resulting in the complete model (C.6) being given as

$$y_k(\boldsymbol{x},\boldsymbol{w}) = f\left(\sum_{j=0}^{M} w_{k,j}^{(2)} h_1\left(\sum_{i=0}^{N} w_{j,i}^{(1)} x_i\right)\right), \quad k = 1,2,\dots,K. \tag{C.9}$$

The neural network model is commonly illustrated as in Figure C.1, where the different nodes indicate the different units in the network, as well as their associated weights.

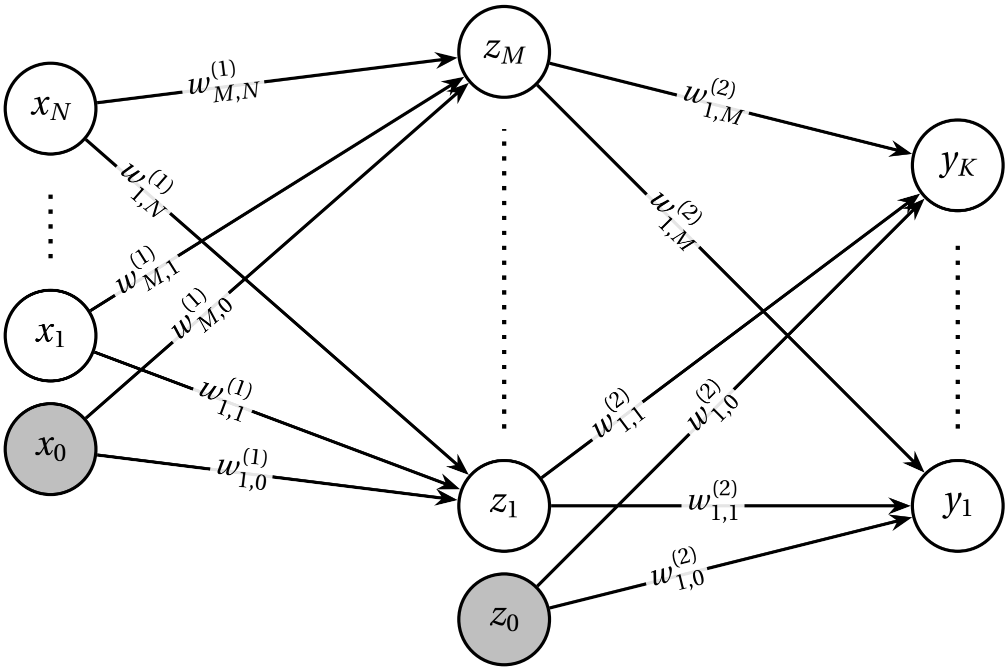


Figure C.1. Illustration of the neural network. Information propagates from left to right. After [Bishop, 2009, Fig. 5.1]

When disregarding the bias parameter, every node from one layer passes information to every node in the next layer; whenever this is the case between two layers, we say they are *fully connected*. While fully connected layers increase the complexity of the network, allowing for more complex interactions between features to be modeled, they introduce a large amount of weights, many of which may be unnecessary. The time needed to fit the model scales with the number of weights, and in many applications, nodes interact mostly with local nodes, having little, if any, relations to distant nodes. *Convolutional neural networks* handle these cases in a more computationally effecient manner and will be introduced in Appendix C.2. Further discussion on these topics requires some elaboration on the topic of activation functions.

C.1.1 Activation Functions

As mentioned briefly earlier in this section, the choice of activation function $h_1(\cdot)$ in (C.3) and output activation function $f(\cdot)$ from (C.5) are dependent on the application. For reasons that will be made clear in Appendix C.1.3, these functions are required to be differentiable. In this subsection, three commonly used activation functions will be introduced; one used in the hidden layers of the network and two output activation functions used in classification.

The *rectifying linear unit* (abbr. ReLU) function is an activation function commonly applied in the hidden layers of a neural network. It is defined as follows:

$$h_{ReLU}(a) = \max\{0, a\} \tag{C.10}$$

Some benefits of the ReLU function include that it is computationally efficient due to its simplicity and the fact that it outputs zero for any negative input, which is likely to result in sparse activations in subsequent layers of the network. Note that the ReLU function is not differentiable at zero, which means that it is not technically a differentiable function. In practice, the derivative of the ReLU function at zero is often defined to be either its left derivative, zero, or its right derivative, one. In this case, any numerical errors that may occur as a result of this are assumed negligible.

For classification problems, a probabilistic approach can be used to argue in favor of certain output activation functions. Consider the posterior probability $P(C_j|\boldsymbol{x})$ of a data point $\boldsymbol{x}$ belonging to class C_j. By Bayes' theorem (Theorem A.20), we have

$$P(C_j|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_j)P(C_j)}{\sum_{k=1}^{K} P(\boldsymbol{x}|C_k)P(C_k)}, \quad j = 1, 2, \ldots, K. \tag{C.11}$$

For multiple binary classifications, we have that $\boldsymbol{x} \in \{C_1, C_2\}$ with $C_1 \cap C_2 = \emptyset$ and we can write (C.11) for class C_1 as [Bishop, 2009, Eq. (4.57)]

$$\begin{aligned} P(C_1|\boldsymbol{x}) &= \frac{P(\boldsymbol{x}|C_1)P(C_1)}{P(\boldsymbol{x}|C_1)P(C_1) + P(\boldsymbol{x}|C_2)P(C_2)} \\ &= \frac{1}{1 + P(\boldsymbol{x})P(C_2)\left(P(\boldsymbol{x}|C_1)P(C_1)\right)^{-1}} \\ &= \frac{1}{1 + e^{-a}}, \end{aligned} \tag{C.12}$$

in which

$$a = \ln\left(\frac{P(\boldsymbol{x}|C_1)P(C_1)}{P(\boldsymbol{x}|C_2)P(C_2)}\right).$$

The function in (C.12) is the logistic sigmoid. When used as an output activation function on an activation a_k, we write it as follows [Bishop, 2009, Eq. (5.6)]:

$$f_{sigm}(a_k) = \frac{1}{1 + e^{-a_k}}. \tag{C.13}$$

Due to its ties to the posterior probability $P(C_k|\boldsymbol{x})$, the output of a neural network using the logistic sigmoid as an output activation function can heuristically be considered the probability that the input $\boldsymbol{x}$ belongs to class C_k.

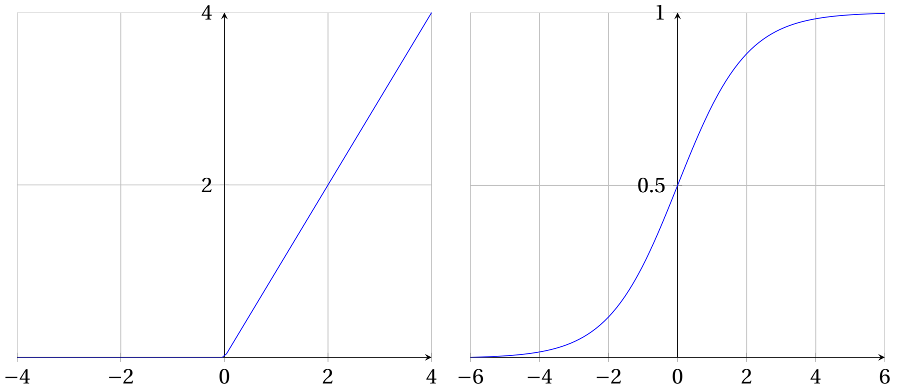


Figure C.2. Examples of activation functions include the Rectifying Linear Unit (ReLU, left) and the logistic sigmoid (right).

For classification problems with a number of classes K greater than two, Bayes' formula (C.11) can be written as [Bishop, 2009, Eq. (4.62)]

$$\begin{aligned} P(C_j|\boldsymbol{x}) &= \frac{P(\boldsymbol{x}|C_j)P(C_j)}{\sum_{k=1}^{K} P(\boldsymbol{x}|C_k)P(C_k)} \\ &= \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}} \end{aligned} \tag{C.14}$$

where

$$a_k = \log\left(P(\boldsymbol{x}|C_k)\right), \quad k = 1, 2, \ldots, K.$$

If taken as a function of a_j, (C.14) is known as the *softmax* function. When used as an output activation function, we write it as follows:

$$h_{smax}(a_k) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} \tag{C.15}$$

C.1.2 Gradient Descent Methods

The final fundamental topic in the theory of feed-forward networks that will be adressed in this section is how the weights $\boldsymbol{w}$ of (C.9) are adjusted through the process of *training*. The aim of the training is to minimize the error (often the sum of squares) of the model (C.9) based on example data. Due to the large number of parameters, neural networks do not have closed-form solutions, and the weights are iteratively updated using numerical methods, with the initial set of weights generated by a pseudorandom number generator.

A commonly applied and simple approach to updating the weights is the *gradient descent* approach, in which the weights $\boldsymbol{w}$ are updated iteratively by taking a step of length η in the negative gradient of the error function taken over the entire set of example data, which we denote $E(\boldsymbol{w})$. Formally, given a fixed $\eta > 0$, we obtain the weights at time $\tau + 1$ from the weights at time τ as [Bishop, 2009, Eq. (5.41)]

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E(\boldsymbol{w}^{(\tau)}), \tag{C.16}$$

where superscript is used to denote iteration step. By use of this method, the weights are updated at each step in the direction of greatest rate of decrease of the error function.

The error function used in (C.16) is taken over all data points in the set of example data before the weights are updated, which results in the training being a slow process. Alternatively, by separating the data set into disjoint subsets that are assumed independent, the error function of the data set can be written as a sum of terms [Bishop, 2009, Eq. (5.42)]

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}), \tag{C.17}$$

where $E_n(\boldsymbol{w})$ denotes the error function taken over the nth subset of data. By applying the gradient descent method (C.16) with randomly selected subsets of the data set rather than the entire data set, we obtain the *stochastic gradient descent* method given by [Bishop, 2009, Eq. (5.43)]

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_n(\boldsymbol{w}^{(\tau)}), \quad 1 \le n \le N, \tag{C.18}$$

which has been found to better deal with redundancy in the data set and to more easily escape local minima [Bishop, 2009, p. 241].

C.1.3 Error Backpropagation

For the stochastic gradient descent method (C.18) to be applicable in practical applications, $\nabla E_n(\boldsymbol{w})$ must be able to be evaluated in an efficient manner. For feed-forward networks, a technique called *error backpropagation* is used for this exact purpose.

The task of evaluating $\nabla E_n(\boldsymbol{x})$ comes down to evaluating each element of the gradient, i.e. each partial derivative $\frac{\partial E_n}{\partial w_{j,i}^{(l)}}$, where $w_{j,i}^{(l)}$ is an element of $\boldsymbol{w}$. While there are multiple possible choices for the error function E_n, a popular candidate is the sum of squares, which is given as [Bishop, 2009, Eq. (5.46)]

$$E_n = \frac{1}{2} \sum_{k=1}^{K} \left(y_{n;k} - t_{n;k}\right)^2, \tag{C.19}$$

where $\boldsymbol{y}_n$ is the vector containing the outputs of the neural network applied to the nth subset of the data and $\boldsymbol{t}_n$ is the corresponding set of labeled data, i.e. the "true values" that we want the model to predict. For the remainder of this section, the subscript n is omitted for ease of notation.

Consider a neural network consisting of L layers with output activation function $f(\cdot)$. We use N_l to denote the number of neurons in layer l. In error backpropagation, we start by considering the partial derivatives $\frac{\partial E_n}{\partial w_{k,j}^{(L)}}$ for the weights $w_{k,j}^{(L)}$ connecting the $(L-1)$th layer to the output layer. Recalling that the outputs of the neural network are given as

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = f(a_k^{(L)}), \quad a_k^{(L)} = \sum_{j=1}^{M} w_{k,j}^{(L)} z_j^{(L-1)}, \quad k = 1, 2, \ldots, N_L, \tag{C.20}$$

we see that he chain rule can be applied [Bishop, 2009, Eq. (5.50)]:

$$\frac{\partial E}{\partial w_{k,j}^{(L)}} = \frac{\partial E}{\partial a_k^{(L)}} \frac{\partial a_k^{(L)}}{\partial w_{k,j}^{(L)}}, \quad k = 1,2,\ldots,K, \quad j = 1,2,\ldots,N_{L-1}. \tag{C.21}$$

From (C.20) we see that

$$\frac{\partial a_k^{(L)}}{w_{k,j}^{(L)}} = z_j, \quad j = 1,2,\ldots,N_{L-1} \tag{C.22}$$

Introducing the notation [Bishop, 2009, Eq. (5.51)]

$$\delta_k^{(L)} = \frac{\partial E}{\partial a_k^{(L)}}, \quad k = 1,2,\ldots,N_L \tag{C.23}$$

which are referred to as *errors* in the context of error backpropagation and should not be confused with the error function. Due to the requirement that the activation function $f(\cdot)$ is differentiable, these errors can easily be evaluated for layer L. In case the error function is the sum of squares (C.19), the error (C.23) is

$$\delta_k^{(L)} = \frac{\partial}{\partial a_k^{(L)}} \left(\frac{1}{2} \sum_{k=1}^{N_L} \left(f(a_k^{(L)}) - t_k \right)^2 \right) = \left(f(a_k^{(L)} - t_k \right) f'(a_k^{(L)}), \quad k = 1,2,\ldots,N_L,$$

where $f'(a_k^{(L)})$ denotes the derivative of $f(\cdot)$ evaluated at $a_k^{(L)}$.

Next, we turn our attention to the evaluation of $\frac{\partial E}{\partial w_{j,i}^{(L-1)}}$, where $w_{j,i}^{(L-1)}$ is a in the $(L-1)$th layer. Recall the activations a_i given (C.7) as well as the hidden units z_i given in (C.3). For layer $(L-1)$ of the network, they can be stated as

$$z_j^{(L-1)} = h_{L-1}\left(a_j^{(L-1)}\right), \quad a_j^{(L-1)} = \sum_{i=1}^{N_{L-2}} w_{j,i}^{(L-1)} z_i^{(L-2)}, \quad j = 1,2,\ldots,N_{L-1}. \tag{C.24}$$

By an approach identical to that of equations (C.21), (C.22) and (C.23), we see that $\frac{\partial E}{\partial w_{j,i}^{(L-1)}}$ can be expressed as

$$\frac{\partial E}{\partial w_{j,i}^{L-1}} = \frac{\partial E}{\partial a_j^{(L-1)}} \frac{\partial a_j^{(L-1)}}{\partial w_{j,i}^{(L-1)}} = \delta_j^{(L-1)} z_i^{(L-2)}, \quad j = 1,2,\ldots,N_{L-1}, \quad i = 1,2,\ldots,N_{L-2},$$

in which the error $\delta_j^{(L-1)}$ can be written as

$$\begin{aligned} \delta_j^{(L-1)} = \frac{\partial E}{\partial a_j^{(L-1)}} &\overset{(a)}{=} \sum_{k=1}^{N_L} \frac{\partial E}{\partial a_k^{(L)}} \frac{\partial a_k^{(L)}}{\partial a_j^{(L-1)}} \\ &\overset{(b)}{=} \sum_{k=1}^{N_L} \frac{\partial a_k^{(L)}}{\partial z_j^{(L-1)}} \frac{\partial z_j^{(L-1)}}{\partial a_j^{(L-1)}} \delta_k^{(L)} \overset{(c)}{=} \frac{\partial z_j^{(L-1)}}{\partial a_j^{(L-1)}} \sum_{k=1}^{N_L} w_{k,j}^{(L)} \delta_k^{(L)} \\ &\overset{(d)}{=} h'_{L-1}\left(a_j^{(L-1)}\right) \sum_{k=1}^{N_L} w_{k,j}^{(L)} \delta_k^{(L)}, \quad j = 1,2,\ldots,N_{L-1} \end{aligned} \tag{C.25}$$

where $h'_{L-1}\left(a_j^{(L-1)}\right)$ denotes the derivative of $h_{L-1}(\cdot)$ evaluated at $a_j^{(L-1)}$. In the equations above, (a) follows from application of the chain rule, (b) follows from another application of the chain rule and from (C.23), (c) follows from (C.20) and (d) follows from (C.24). We see from (C.25) that by passing error information $(\delta_k^{(L)})$ 'backwards' through the network, we obtain a simple method of evaluating the error at each neuron of the network and - by application of (C.21) - evaluating each entry of ∇E_n. This is the main result of error backpropagation; by utilizing knowledge of the errors of layer $l+1$, the errors of layer l can be efficiently computed.

C.2 Convolutional Neural Networks

The aim of this section is to introduce *convolutional neural networks* (abbr. CNN), a class of neural networks which are invariant to certain transformations of the input data. Convolutional neural networks are often applied to classification of image data, where the identity of the input data is invariant under translations, scaling and small rotations. Using classification of digital image data as an example, one could simply construct a simple feed-forward network as discussed in Appendix C.1 with the pixel intensities as inputs and the softmax function (C.15) as an output activation function. However, such an approach ignores the fact that nearby pixels are more strongly correlated than distant pixels [Bishop, 2009, Sec. 5.5.6].

Convolutional neural networks make use of the convolution operator in place of a general matrix product in at least one of their layers in order to extract local features rather than global features [Goodfellow et al., 2016, Ch. 9]. How exactly this changes the model (C.9) will be the main topic of Appendix C.2.1. Additionally, most convolutional networks also apply *pooling* in order to reduce variability. This operation will be introduced in Appendix C.2.2.

C.2.1 Convolutional Layer

In order to introduce the concept of convolutional layers, it is first necessary to understand the discrete convolution operator.

Definition C.1 (Discrete Convolution)
The discrete convolution of two sequences $x_n \in \mathbb{R}$ and $w_n \in \mathbb{R}$, denoted $(x * w)_n$ is defined as

$$(x * w)_n = \sum_{k\in\mathbb{Z}} x_k w_{n-k}, \tag{C.26}$$

provided that the sum exists. ▲

In the context of neural networks, the signal x_n is usually a multidimensional array of data and the sequence w_n is called a *kernel*. The position of the two within the convolution can be interchanged freely, since the convolution is a commutative operation.

Lemma C.2
The discrete convolution is commutative, i.e. $(x * w)_n = (w * x)_n$. ▲

Proof

By introducing a substitute variable $\hat{k} = n - k$, we see that

$$(x * w)_n = \sum_{k \in \mathbb{Z}} x_k w_{n-k} = \sum_{\hat{k} \in \mathbb{Z}} x_{n-\hat{k}} w_{\hat{k}} = (w * x)_n$$

■

While the commutative property might be useful in other applications, it is generally not an important property of neural network implementations [Goodfellow et al., 2016, Ch. 9.1]. Instead, by "flipping" the kernel w_n relative to the input x_n, such that their indices both increase as in (C.26), we obtain the *cross-correlation sequence*. It is defined as follows:

Definition C.3 (Cross-Correlation)

The (1-dimensional) cross-correlation between sequences x_n and w_n is defined as [Goodfellow et al., 2016, Eq. (9.6)]

$$S_{x,w;n} = \sum_{k \in \mathbb{Z}} x_{n+k} w_k$$

provided that the sum exists. ▲

The discrete convolution and cross-correlation can be seen as a multiplication of the input by a matrix in which several entries are equal to other entries. More specifically, the matrix is a Toeplitz matrix, i.e. every row of the matrix is equal to the row above shifted by one element.

In order to apply the concept of cross-correlation sequences to neural networks, one may begin by noting that the input vector $\boldsymbol{x}$ has a finite number of entries. Thus, the kernel is similarly restricted to having a finite number of entries, with this number being smaller than that of the input. This allows for the output of the cross-correlation function to provide information about local features of the input. The output is commonly restricted to positions for which the kernel is contained within the input.

Next, recall that the activations in the simple feed forward network were given by (C.7), i.e.

$$a_j^{(1)} = \sum_{i=0}^{N} w_{j,i}^{(1)} x_i, \quad j = 1, 2, \ldots, M.$$

We see that with an input vector of length N, the network has $N+1$ weights (one for each input and an additional bias parameter) for each of the M outputs, resulting in a total of $(N+1)M$ weights which would have to be adjusted through training. On the other hand, if we were to relate the input and activation by a cross-correlation function as defined in Definition C.3 between the input and a sequence of weights $w_n, n = 1, 2, \ldots, N_w$ of length $N_w < N$, we would have

$$a_j^{(1)} = \sum_{i=1}^{N_w} x_{j+i-1} w_i + w_{0,j}, \quad j = 1, 2, \ldots, N - N_w + 1, \tag{C.27}$$

in which case the weights are shared between different activations. We call a layer in which the activations are on the form of (C.27) a *convolutional layer* and disregard the fact that a more technically correct name would be a cross-correlation layer. We call any neural network that includes one or more convolutional layers a *convolutional neural network* (abbr. CNN).

There are some benefits to using CNNs over simple feed-forward networks. Due to the weights being shared between activations, there are fewer parameters that require fitting. As a result, the training should be faster and require less storage. Additionally, while each input in a traditional neural network layer interacts with each output, convolutional neural networks have sparse interactions, which improves the efficiency of the model [Goodfellow et al., 2016]. While a traditional neural network layer with N inputs and M outputs require a matrix multiplication with $O(N \cdot M)$ operations, each of the $M = N - N_w + 1$ outputs of a convolutional layer interact with only N_w inputs (N_w being the length of the kernel), resulting in a $O(N_w \cdot M)$ operations. Provided that N_w is several orders of magnitude smaller than N, this can yield a significant reduction in runtime [Goodfellow et al., 2016].

While the error backpropagation method cannot be directly applied in the same manner as shown earlier when dealing with backpropagation through convolutional layers, some small adjustments can be applied to (C.25) to make error backpropagation applicable again. Consider a network, whose output layer (L) is a convolutional layer in which the output activations $a_k^{(L)}$ are given by a model on the form (C.27) with a kernel of length N_w. The choice of output activation function is unimportant. We seek to determine the partial derivative of the error function with respect to an activation in layer $(L-1)$ and begin by considering equation (a) from (C.25):

$$\begin{aligned} \delta_j^{(L-1)} &= \sum_{k=0}^{N_L-1} \frac{\partial a_k^{(L)}}{\partial z_j^{(L-1)}} \frac{\partial z_j^{(L-1)}}{\partial a_j^{(L-1)}} \delta_k^{(L)} \\ &\overset{(a)}{=} \frac{\partial z_j^{(L-1)}}{\partial a_j^{(L-1)}} \sum_{k=0}^{N_L-1} \delta_k^{(L)} \frac{\partial}{\partial z_j^{(L-1)}} \left(\sum_{i=1}^{N_w} z_{k+i}^{(L-1)} w_i^{(L)} + w_{0,k}^{(L)} \right) \\ &\overset{(b)}{=} \frac{\partial z_j^{(L-1)}}{\partial a_j^{(L-1)}} \sum_{k=0}^{N_L-1} \delta_k^{(L)} w_{j-k} \mathbf{1}_{[0,N_w-1]}(j-k), \end{aligned} \tag{C.28}$$

where $\mathbf{1}_{[0,N_w-1]}(\cdot)$ is used to denote the indicator function that is one when the argument is in the interval $[0, N_w - 1]$ and zero otherwise. In the equations above, (a) follows from the fact that the activations $a_k^{(L)}$ follow a model on the form (C.27), and (b) follows from noting that the jth entry of $z_j^{(L-1)}$ appears in the rightmost sum only if $j = k + i$ is in the interval $[0, N_L - 1]$ - if it is, the jth entry is multiplied by the weight $w_{j-k}^{(L)}$.

In many applications, we may want to apply cross-correlation to multi-dimensional signals. In particular, when dealing with data arranged as a 2-dimensional grid such as images, the cross-correlation is a 2-dimensional sequence defined as follows:

Definition C.4 (2-Dimensional Cross-Correlation)

The 2-dimensional cross-correlation between sequences $x_{m,n} \in \mathbb{R}$ and $w_{m,n} \in \mathbb{R}$, denoted $S_{x,w}(m,n)$, is defined as [Goodfellow et al., 2016, Eq. (9.6)]

$$S_{x,w}(m,n) = \sum_{i,j\in\mathbb{Z}} x_{m+i,n+j}\, w_{i,j}, \tag{C.29}$$

provided that the sum exists. ▲

The analogue to (C.27) for 2-dimensional inputs with a kernel matrix of size $N_1 \times N_2$ is given as

$$a^{(1)}_{m,n} = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} x_{m+i-1,n+j-1}\, w_{i,j}, \quad j = 1,2,\ldots,M,$$

An example of a cross-correlation is shown in Figure C.3.

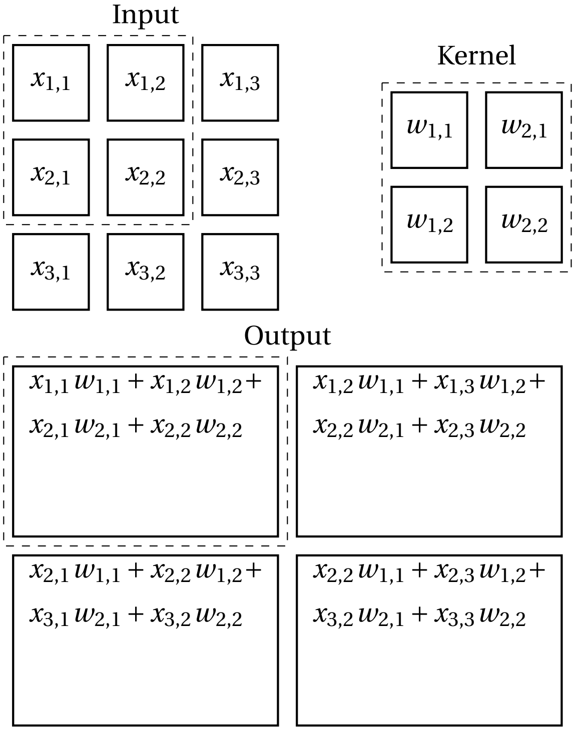

Input
$x_{1,1}$
$x_{1,2}$
$x_{1,3}$
$x_{2,1}$
$x_{2,2}$
$x_{2,3}$
$x_{3,1}$
$x_{3,2}$
$x_{3,3}$
Kernel
$w_{1,1}$
$w_{2,1}$
$w_{1,2}$
$w_{2,2}$
Output
$x_{1,1}w_{1,1} + x_{1,2}w_{1,2} + x_{2,1}w_{2,1} + x_{2,2}w_{2,2}$
$x_{1,2}w_{1,1} + x_{1,3}w_{1,2} + x_{2,2}w_{2,1} + x_{2,3}w_{2,2}$
$x_{2,1}w_{1,1} + x_{2,2}w_{1,2} + x_{3,1}w_{2,1} + x_{3,2}w_{2,2}$
$x_{2,2}w_{1,1} + x_{2,3}w_{1,2} + x_{3,2}w_{2,1} + x_{3,3}w_{2,2}$

Figure C.3. Illustration of a 2-D cross-correlation on matrix form. The dashed squares indicates the elements of the input and kernel from which the upper-left entry of the output is formed. After [Goodfellow et al., 2016, Fig. 9.1]

C.2.2 Pooling

Convolutional networks typically employ convolutional layers as part of a three-stage process [Goodfellow et al., 2016]. First, several cross-correlations are performed in parallel, yielding a set of activations. In the second stage, a nonlinear activation function such as the ReLU function mentioned in Appendix C.1.1 is applied to each of the activations. In the final stage, the output of the layer is further modified by the use of a *pooling function*, which replaces the output of the layer with a summary statistic of local outputs [Goodfellow et al., 2016]. Examples of such summary statistics include taking local averages or the L^2-norm (Definition A.1). The summary statistic of greatest interest to this project, however, is the maximum, whose associated pooling function is called *max pooling*.

For each input, the max pooling operator outputs the maximum input value from a region around the input. This is illustrated in Figure C.4. Max pooling can help make the model approximately invariant to small translations, which is a useful property if the presence of a feature in the input is more important than its exact location [Goodfellow et al., 2016]. An example of this is shown in Figure C.4, in which the inputs to the bottom example are those of the top example shifted by one unit. Despite every input being shifted by one unit, only three of the eight outputs differ between the two examples.

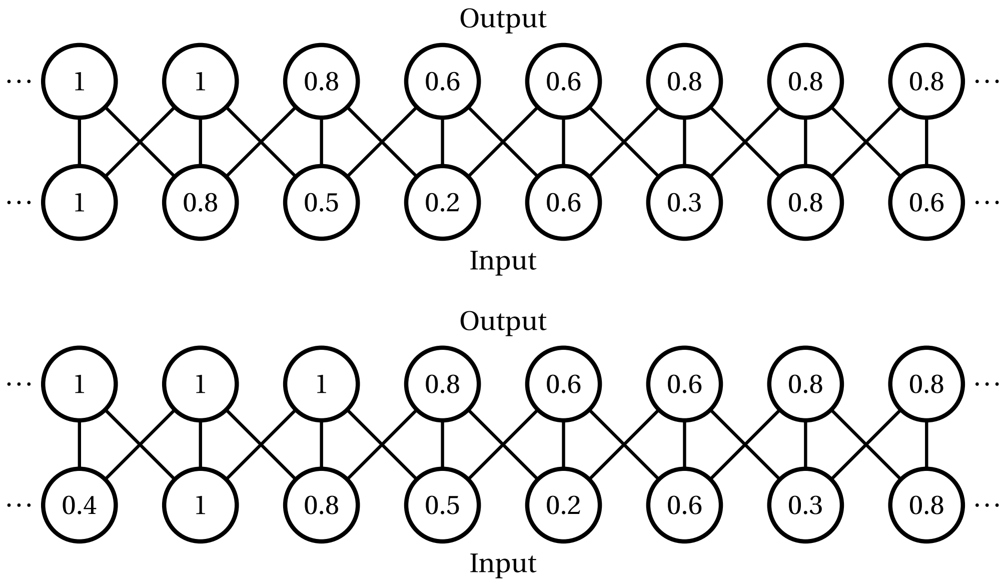


Figure C.4. Illustration of max pooling. Lines are used to indicate which inputs interact with which outputs. The input units in the bottom figure are the same as those of the top figure shifted by one unit to the right. After [Goodfellow et al., 2016, Fig. 9.8]

.

Max pooling can also be used to reduce the number of units in the network by simply outputting the maximum in the neighborhood of every k input units rather than every input unit, resulting in the next layer having approximately $1/k$ times the number of units of the previous layer [Goodfellow et al., 2016]. This can be used to increase the computational efficiency of the network.