Statistical and machine learning methods for classification of episodic memory

Damir Basic Knezevic and Albin Heimerson

Master's thesis
2018:E32

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum scientiarum mathematicarum

Master's Theses in Mathematical Sciences 2018:E32
ISSN 1404-6342

LUTFMS-3346-2018

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/

Abstract

Multiple modern methods of statistical feature extraction and machine learning are applied to classification of encoding and retrieval of episodic memories using electroencephalogram (EEG) recordings. Raw data, different time-frequency methods, and multiclass common spatial patterns are used for statistical feature extraction. For each type of feature extraction multiple machine learning algorithms are tested and compared. Classification accuracies of up to 82 % are reached with one-dimensional convolutional neural networks on raw data. It is found that more complex and time-consuming classifiers generally improve the accuracy. However, the features chosen are the main factor deciding the accuracy. A novel idea for designing an encoding-retrieval classifier is discussed and implemented. In spite of having multiple different designs, almost all classifier combinations involving the retrieval data fail to reach significant classification levels.

Acknowledgements

Thanks to Mikael Johansson at the Department of Psychology, Bo Bernhardsson at Department of Automatic Control and Maria Sandsten at the Division of Mathematical Statistics for all their help and support.

Thanks also to Mattias Ohlsson at the Department of Computational Biology and Biological Physics, Johan Lindström at the Division of Mathematical Statistics, and Johan Eker at the Department of Automatic Control for valuable discussion and inspiration.

Contents

1

Introduction

1.1 Classification of memories

The human body is immensely complex, and in its processing centre lies the arguably most complex body part: the brain. We humans know more about galaxies millions of light years away and particles a hundred million times smaller than the wavelength of visible light than we do about the inner workings of the brain, but this is not due to a lack of trying. From physicists trying to accurately model and predict brain activity based on the electromagnetic pulses that constantly flow through the neurons, to psychiatrists using behavioural patterns to treat psychological illnesses and to improve the daily lives of many, there is no shortage of academics working on understanding the brain. Somewhere between these extremes lies the field of cognitive neuroscience, where the aim is to understand the specific neural mechanisms by which mental processes occur. Being grounded in, among other topics, experimental psychology, neurobiology, mathematics, and brain imaging techniques, it is a very interdisciplinary field.

It is well known that some regions of the brain are responsible for some specific types of activities. For example, the visual cortex is located in the very back of the head [C. Van Essen et al., 1992], and Wernicke's area, one of the areas commonly associated with understanding speech, is located toward the back of the dominant cerebral hemisphere (the left one in most right-handed individuals) [Gupta, 2014]. Different patterns of behaviour are said to be *generated* in certain parts of the brain [Freeman, 1975], and research within the field of cognitive neuroscience is often conducted using various imaging techniques in order to explore whether certain parts of the brain are activated more than others when exposed to certain stimuli. Two very common imaging techniques are electroencephalography (EEG) and functional magnetic resonance imaging (fMRI).

EEG works by placing a set of electrodes, sometimes in a cap, on the scalp and recording the voltage over the electrodes. Figure 1.1 show the electrode placement used for our data and Figure 1.2 show a subject using an EEG cap. The working assumption with this type of imaging is that measured voltage changes correlate with heightened or altered activity. fMRI works by placing the subject in a very large ma-

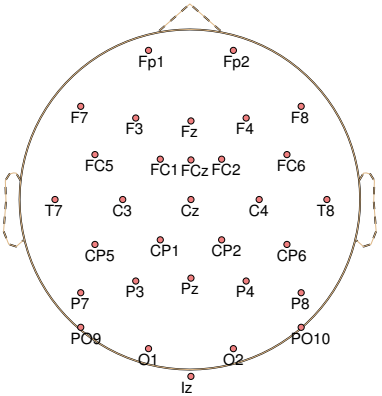

Figure 1.1 Locations of the sensors in the EEG cap.

Figure 1.2 An EEG cap with electrodes on a subject.

chine and asking them to lay very still while the machine measures the blood flow in the brain. The working assumption when analysing fMRI data is that cognitive activity correlates with a change in blood flow in that region [Dekker and Sijbers, 2005]. The main advantages of EEG over fMRI are that it is cheaper and has much better time resolution—the voltages can be captured with a sampling time in the order of milliseconds even with relatively inexpensive setups. fMRI machines typically cost more both to buy and to operate, and images are typically recorded every 1–10 seconds. fMRI, on the other hand, has much higher spatial resolution [Lystad and Pollard, 2009]. Hundreds or thousands of pixels per image is not uncommon,

while EEG setups typically have around 20–40 electrodes (although *dense* EEG clusters of 128, 256, and up to 512 electrodes have been devised recently) [Ryynanen et al., 2004]. Another disadvantage of EEG is that imaging is essentially limited to the outermost parts of the brain—the cortex—while fMRI directly displays activity anywhere in the brain, three-dimensionally versus the two-dimensional surface potential of EEG. Last but certainly not least, the signal-to-noise ratio is infamously bad for EEG recordings, requiring careful pre-processing to remove signal artefacts stemming from e.g. eye movement and blinking to get representative and usable data [Ihalainen et al., 2015]. Nevertheless, the excellent temporal resolution and inexpensive equipment are enough to encourage further use of the technology.

One practical issue with trying to analyse data from neurocognition experiments, whether it be EEG or fMRI (or any other imaging technique) is that the experiments are typically conducted by academics extremely specialised within psychology, with surface knowledge about statistics, signal processing, and machine learning. As such, without an interdisciplinary drive, analysing the data can be quite a cumbersome task. After reading a great many papers on computational neuroscience, the psychologists designing the experiment often seem to have an idea in mind about what *should* be happening in the brain, but easily and readily interpretable data is an exception rather than a rule. Very often some sort of all-encompassing toolbox (like FieldTrip for MATLAB [Oostenveld et al., 2011] or MNE for Python [Larson et al., 2016]) is used. The toolboxes often contain extremely advanced and state-of-the-art techniques for feature extraction and classification which very often have high accuracy, but are not very easy to interpret. It is comparably easy to choose features and classifiers based on what gives a high classification accuracy, rather than designing a feature extraction scheme based on models and existing theory. If the aforementioned scheme fails and something entirely different produces much more appealing results, it can be tempting to use the more succesful scheme and try to explain the results afterwards. This type of investigation has its merits, but for finding neural generators and furthering neurocognitive research, the model-driven approach seems more useful. For us engineering students interested in interdisciplinary research, a more interesting approach is for the psychologist gathering the data to present their theory to the statistician or data expert, who then converts the abstract concept into an interpretable feature extraction algorithm that is directly testable.

1.2 The experiment

The experiment from which the data is gathered was conducted by Inês Bramão and Mikael Johansson at the Department of Psychology in Lund. A more thorough description can be found in Section 4.1, but the paradigm is essentially this: during the *encoding phase*, test subjects are exposed to an image from one of three classes—faces of famous people, famous landmarks such as the Niagara Falls or

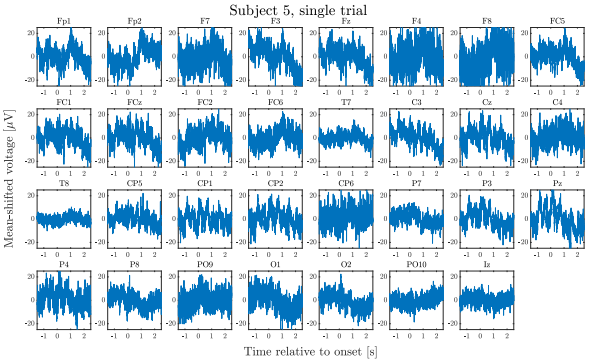

Figure 1.3 One trial (31 channels, 4 seconds) sampled to visualise the data. Each channel corresponds to the data recorded from one sensor. The samples are, without loss of generality, from a random trial of subject 5.

the Big Ben, or man-made objects such as a violin or a hammer—and we are to use statistical feature extraction and machine learning techniques to predict, from data, which type of image the subject was exposed to. It should be noted that the subjects were not instructed to think of the category, but instead they were shown a word that they were to couple to the image as a word-image pair. Later on, during the *retrieval phase*, the subject was shown the same word and was instructed to recall the image associated to that word in their head. The idea is that images from the same categories should activate the same neural generators in the brain in a process called *ecphory* [Waldhauser et al., 2016], and that this should serve as the main type of feature that separates the classes. Our main aim, after (hopefully) successfully classifying the images seen during the encoding phase, is to use the same algorithm to classify on the retrieval phase with minimal or no alterations. If we succeed in designing such a classifier, the features it extracts can be analysed with the aim of finding the link between encoding and retrieval in a physical and psychological framework. In Figure 1.3 we show a sample of what the data can look like.

1.3 A broader perspective

The aim of the experiment is mainly to see if EEG data, using machine learning and feature extraction based on known neurocognition theory, is enough to distinguish between three classes of images. Note that this problem is extremely difficult right at the onset—not only are the subjects exposed to similar images at every stimulus, but the three classes are very abstract as well. The theory is solid: apart from the visual cortex being activated, the neural generators for the three different image categories should be different. Also, the categories should activate generators on different time scales which means that there are some temporal feature extraction

possibilities there as well. However, the slightest loss of focus in subjects will show clearly in the EEG, and due to reaction times varying even within subjects during different stimuli, temporal feature extraction is not so straightforward.

When searching for articles related to EEG in journals and conferences related to signal processing, statistics, and machine learning, we mostly came across articles treating brain-computer interfaces, and some connected to anomaly detection and sleep EEG. Brain-computer interfaces (BCI) are, in general terms, computer systems that interact with humans through direct measurements on the brain and predictions made from those. Most of the authors in the BCI category seem to test their algorithms on *simulated movement*, which means letting subjects imagine they are moving e.g. their left or right arm, and the classifier is trained to distinguish between these two paradigms. We also explored anomaly detection but not the sleep EEG application. The former field is concerned with detecting either psychic illnesses like schizophrenia [Laton et al., 2014] or Alzheimer's disease [Gómez et al., 2017], or critical conditions like epileptic seizures [Boashash et al., 2015]. The BCI experiments with simulated movement are isolated, simple classification tasks with a clear left-right asymmetry [Rogers et al., 2004] where no level of abstraction is needed from the classifier. Seizure detection is more a question of automation than of developing novel features, since a visual inspection of EEG data shows rather clearly whether the subject has experienced a seizure or not. Illness detection is arguably harder and not so clear-cut, but is nevertheless a question of distinguishing normal from abnormal. In the experiment central to this thesis, the three classes are extremely similar and it is completely impossible to see visually from the EEG data to which class the data belongs.

As engineering students, we are entering mostly uncharted territory. The problem at hand is extremely difficult, but nevertheless important to develop answers to. Extracting features related to activation of neural generators could perhaps be applied directly to BCI or illness detection as a general feature extraction method, as long as the theory of generators in cognition holds. We also firmly believe that promoting and encouraging interdisciplinary research and cooperation is beneficial to all parties.

1.4 Previous analysis of this data set

The experiment was carried out and has been analysed by Bramão and Johansson [Bramão and Johansson, Submitted for publication, 2018], but an even earlier version of a similar paradigm was carried out in 2005 [Polyn et al., 2005]. In Polyn's paper, the data is collected using fMRI with a sampling frequency of around 0.56 Hz (one sample every 1.8 seconds), and the motivation for the later paper was that classification is supposed to be faster than that. Indeed, they found evidence of classification between categories after no more than about 170 milliseconds, during the early perceptual stages of encoding. Both studies, however, found evidence of re-

trieval taking a longer time than encoding, and classifying power becoming evident significantly later for the retrieval stage of the experiment.

The same data that we use has been analysed in a course project at Lund University, a Bachelor's Thesis [Dalin-Volsing, 2015], a Master's Thesis [Heyden, 2016], a Licentiate Thesis [Anderson, 2017], and of course by Bramão and Johansson. The work done by Dalin-Volsing is quite limited in scope and aims to use a certain type of multitaper spectrogram (see Section 2.1) as the only feature extraction method. Heyden limits himself to the raw data and uses analysis of variance (ANOVA) methods to reduce the number of features and then uses quite a simple type of classifier to distinguish the three classes. Anderson aims to derive optimal covariance estimates with the assumption that the EEG is a locally stationary process. In [Bramão and Johansson, Submitted for publication, 2018], the authors use the wavelet transform channel-wise on the EEG data to extract features, perform feature reduction using ANOVA and then classify with a support vector machine, a last step similar to Heyden's work.

1.5 Contribution from this work

As we see this project, it is split up into two fairly distinct parts: pre-processing and feature extraction, and classification. We aim to explore both aspects of the problem, and propose novel ideas in both areas based on experience gathered during our studies.

For pre-processing and feature extraction, we try to seek inspiration from the two engineering-heavy EEG analysing fields that we have come across: abnormality detection and BCI.

The researchers working on improving abnormality methods typically use time-frequency methods, where the frequency components of a signal are attempted to be visualised over time. It has been suggested that this is a natural course of action since EEG signals are non-stationary, and thus their frequency characteristics change over time [Boashash et al., 2015]. The wavelet transform [Daubechies, 2006] appears regularly in articles both in signal processing journals and neurophysiology journals. The classification step is often some type of statistical feature extraction step [Boashash and Ouelha, 2018], heavy downsampling, and/or ANOVA feature reduction step [Bramão and Johansson, Submitted for publication, 2018], followed by classifying with a simple linear support vector machine (SVM) or something similar. We explore some standard time-frequency techniques, not limiting ourselves to the wavelet transform, and classify them using 2-D convolutional neural networks, which are state-of-the art classifying algorithms that are very popular for images.

BCI researchers often use features extracted from spatial patterns in the EEG. Common algorithms include principal component analysis (PCA), independent component analysis (ICA), and common spatial patterns (CSP) [Jing and Yun, 2017]. We extract features using a multiclass CSP algorithm that turns out to be

equivalent to a type of ICA [Grosse-Wentrup and Buss, 2008], and perform the classifying step using both support vector machines and multilayer perceptron networks. Both of these algorithms are chosen due to CSP outputting a relatively small amount of features compared to the transform images (the order of magnitude being 100 for CSP and about 1 million for the time-frequency transforms), and as such we do not need the same complexity in the classifiers.

1.6 Report outline

Chapter 2 focuses on giving a short mathematical introduction and motivation for time-frequency analysis and CSP. Chapter 3 introduces the machine learning algorithms used in the thesis. Chapter 4 describes the methods used for classifying data and provides parameter details about the tests, the results of which are presented in Chapter 5. Finally, a discussion reflecting on the results and ideas for future investigations is presented in Chapter 6.

2

Feature extraction

2.1 Time-frequency transformations

A standard technique within EEG data feature extraction and classification is to use the frequency information that the temporal data provides. A common way to extract spectral information in a signal is to use the Fourier transform. The Fourier transform of the signal $x(t)$ has a few different definitions all being used in parallel [Boashash, 2013], one of them being

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} \, \mathrm{d}t. \tag{2.1}$$

The resulting function $X(f)$ is in general complex and its modulus and argument describe the power and phase of each frequency in the signal, respectively. In many cases the Fourier transform provides enough information, especially about periodic signals, to perform accurate signal classification (if that is the goal). In the case of EEG data, however, the Fourier transform fails in one key aspect: the signal to be transformed is assumed to be *stationary*. This means that the frequency information in the signal is assumed to stay constant for the entire time it is collected. In the study at hand, where a subject is exposed to a stimulus during the time data is collected, it is very clear that the signal is not stationary, and it is well known that the background EEG itself is not stationary even when subjects' environments remain unchanged [H. Jansen et al., 1981; Pardey et al., 1996; Kaplan et al., 2005]. An illustration of what happens when the non-stationarity of a signal is not taken into account can be seen in Figure 2.1.

A natural next step is to consider time-frequency analysis, a field where the aim is to simultaneously extract both time and frequency information. The main topic which drives the field forward is a version of the well-known uncertainty principle, which states that perfect resolution cannot be attained both in time and frequency simultaneously—there is a compromise. This makes sense because frequency estimation is very closely related to period length. More periods gives better frequency estimation, but also a larger signal length which results in a lower temporal resolution. At its core, time-frequency estimation aims to exploit structure in the data

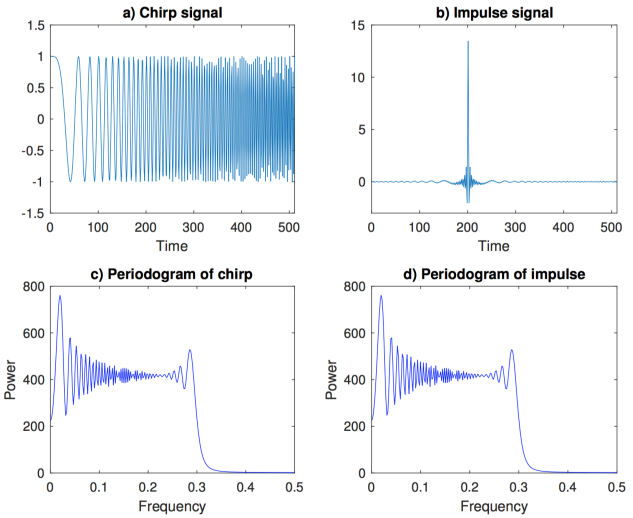

Figure 2.1 Illustration of the shortcomings of the periodogram, $\left|X(f)\right|^2$. **a)** is a plot of a *chirp* signal—a periodic signal with (in this case) linearly increasing frequency—and **b)** shows an impulse. It turns out that the periodograms **c)** and **d)** are identical, because all of the information separating the two transforms is hidden in the phase. The phase, in turn, is rarely shown. The image is borrowed from [Sandsten, 2018] with the author's permission.

to reduce the effect of the compromise. Indeed, time-frequency methods have been fairly widely used to study EEG data but mostly in detection of abnormalities, such as epilepsy or schizophrenia. [Rutkowski et al., 2013; Roach and Mathalon, 2008; Muthuswamy and Thakor, 1998]

Short-time Fourier transform and spectrogram

One way to perform time-frequency analysis is by extracting parts of the time signal and evaluating multiple Fourier transforms. The partial time signal is extracted by multiplication with a *window function* $h(t)$ that is typically symmetric, has compact support, tapers off toward the ends of its support, and is normalised such that $\int_{-\infty}^{\infty}\left|h(t)\right|^2 \mathrm{d}t = 1$. The short-time Fourier transform (STFT) is then defined as [Sandsten, 2018]

$$X(t,f) = \int_{-\infty}^{\infty} x(t_1)h^*(t_1 - t)e^{-i2\pi f t_1}\,\mathrm{d}t_1 \tag{2.2}$$

where * denotes the complex conjugate. The *spectrogram*, which is sometimes seen as a built-in function in certain audio analysis software such as e.g. Audacity and Transcribe!, is simply defined as

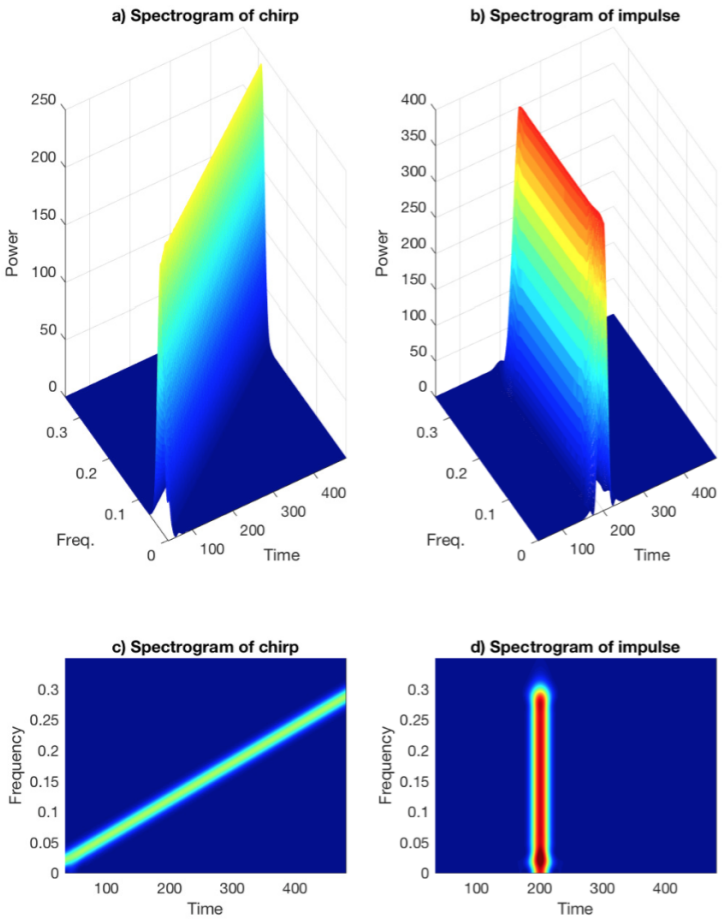

Figure 2.2 These images show the short-time Fourier transform (STFT) of the same signals as in Figure 2.1. Here it is clear, even from the absolute value alone, that the signals have very different frequency content. The image is borrowed from [Sandsten, 2018] with the author's permission.

$$S_x(t,f) = \left|X(t,f)\right|^2. \tag{2.3}$$

The benefits of showing the frequency content as a function of time are illustrated in Figure 2.2, as compared to the shortcomings of the Fourier transform and periodogram in Figure 2.1. The spectrogram is good for visualising many signals and it is simple to implement, which is probably why it is included in for example audio

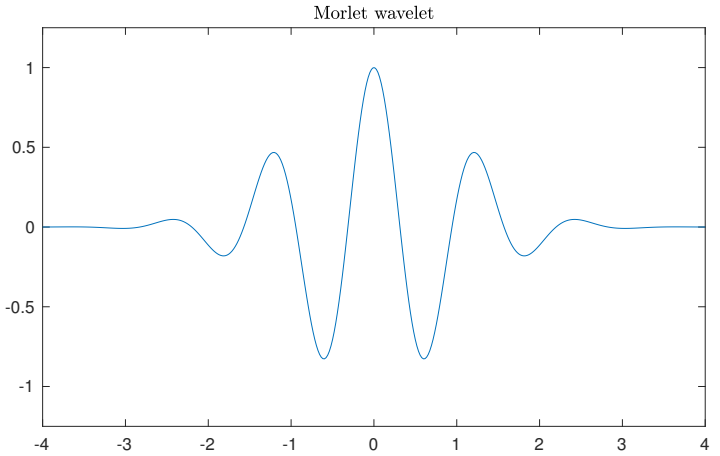

Figure 2.3 Standard Morlet wavelet [Daubechies, 1992].

analysis software, but the volatility to changes in window length and shortcomings in frequency resolution inspire the search for better time-frequency visualisation tools.

Wavelet transform

The wavelet transform (or wavelet decomposition) has been used within the subject of extracting features from EEG for a long time and with considerable success [Hazarika et al., 1997]. The idea behind the wavelet transform is to use a specific type of scaleable window in the STFT. The family of windows is typically defined as

$$h_{a,b}(t) = |a|^{-1/2} \, h\left(\frac{t-b}{a}\right) \tag{2.4}$$

and is typically called the *wavelet* family [Grossmann and Morlet, 1984; Daubechies, 1993]. The definition of the continuous wavelet transform of the signal $x(t)$ is then

$$X_\omega(a,b) = \int_{-\infty}^{\infty} x(t) h^*_{a,b}(t) \, \mathrm{d}t \tag{2.5}$$

where the parameters a and b are interpreted as approximate frequencies and time centers, respectively [Daubechies, 1992]. A common type of wavelet is the *Morlet wavelet*, as illustrated in Figure 2.3, where the *mother wavelet* $h(t)$ is an exponential plane wave windowed by a Gaussian, or

$$h(t) = \left(e^{-ift} - e^{-\frac{1}{2}f^2}\right) e^{-\frac{1}{2}t^2}, \tag{2.6}$$

scaleable and moveable with the constants a and b from (2.4) [Daubechies, 1992] and the frequency f as a constant. In practice, the parameter is usually chosen as

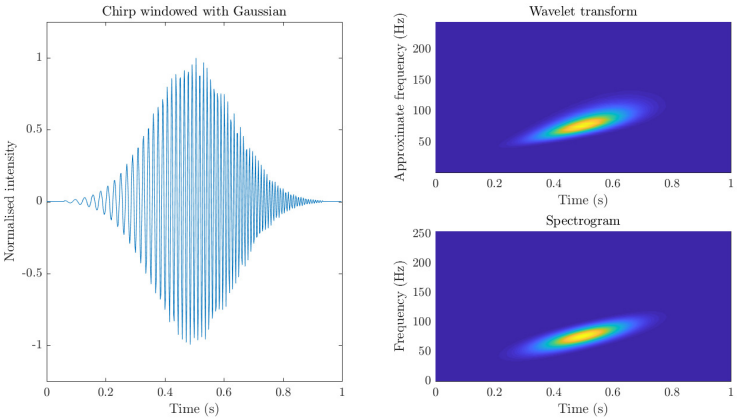

Figure 2.4 Comparison between continuous wavelet transform and spectrogram (Hann window, length 32 samples) for a linear chirp signal ($F_s = 512$) windowed by a Gaussian pulse. While the spectrogram shows an evenness in time and frequency, the wavelet transform is clearly more diffuse at the higher than at the lower frequencies.

$f = 5$ and the imaginary part is largely ignored, so a version of this expression that occurs regularly is

$$h(t) = e^{-\frac{1}{2}t^2}\cos(5t). \tag{2.7}$$

Nevertheless, the idea is that by changing the frequency of the wavelet (while keeping the power constant) and convolving it repeatedly with the signal in time, a measure of the power of those frequencies over time can be attained and shown in a figure. Essentially, the wavelet is a band-pass filter [Mallat, 1989].

A double-edged sword inherent to the wavelet transform is that higher-frequency wavelets (that is, when the scaling parameter a is small) are naturally more narrow and thus have better time resolution and poorer frequency resolution [Daubechies, 2006]. An illustration of this phenomenon can be seen in Figure 2.4, and a schematic of the change in resolution with parameter choice is shown in Figure 2.5. Where the STFT has the spectrogram, the continuous wavelet transform has the *scalogram*. It is similarly defined as

$$S_\omega(a,b) = \left|X_\omega(a,b)\right|^2 \tag{2.8}$$

and the main difference is that the approximate frequencies in the scalogram are on a logarithmic axis instead of a linear one. Unsurprisingly, the wavelet transform is preferred to the STFT for applications where resolving high frequencies well in time is more important than resolving low frequencies. Whether EEG data fits this criterion is up to debate and depends on the application, but it cannot be denied that it is a useful tool for feature extraction.

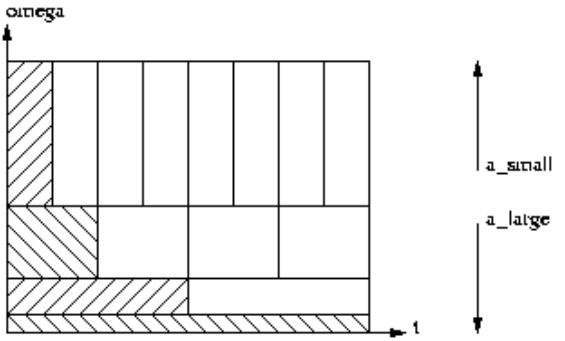

Figure 2.5 A schematic illustrating the trade-off between time and frequency resolution in the wavelet transform. As the value of the scaling parameter a increases, the time resolution decreases while the frequency (or angular frequency, omega) resolution increases, and vice versa. As such, the wavelet transform is well suited for capturing for example transient high-frequency phenomena. Image downloaded from [*Continuous Wavelet Transform.* N.d.]

Wigner distribution

The Wigner distribution serves as a general base tool in time-frequency analysis and is in some sense a generalisation of the spectrogram. The Wigner distribution $W_x(t,f)$ of a signal $x(t)$ is defined as

$$W_x(t,f) = \int_{-\infty}^{\infty} x\big(t+\tfrac{\tau}{2}\big)x^*\big(t-\tfrac{\tau}{2}\big)e^{-i2\pi f\tau}\,\mathrm{d}\tau. \tag{2.9}$$

The factor

$$r_x(t,\tau) = x\big(t+\tfrac{\tau}{2}\big)x^*\big(t-\tfrac{\tau}{2}\big) \tag{2.10}$$

can be identified as the *instantaneous autocorrelation function*, or IAF, so this expression is closely related to the time-varying *power spectral density*

$$S_x(t,f) = \int_{-\infty}^{\infty} E\big[r_x(t,\tau)\big]e^{-i2\pi f\tau}\,\mathrm{d}\tau, \tag{2.11}$$

known from theory on stochastic processes [Lindgren et al., 2014].

Cross-terms The definition of the Wigner distribution contains a product of the signal with itself due to the IAF (disregarding for a moment a time shift and conjugate operator). If the signal x can be expressed as a sum of two (or more) signals, the product will introduce terms that are products between the same signals, called auto-terms, and products between different signals, called cross-terms. Both the auto-terms and the cross-terms as well as a comparison between the Wigner distribution and the spectrogram can be seen in Figure 2.6. Since real life signals most

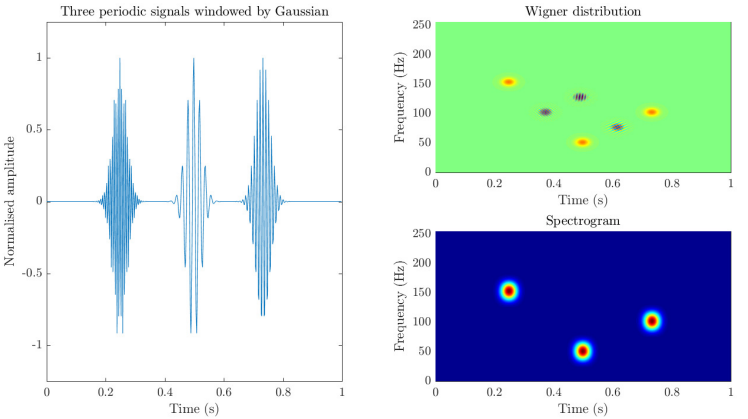

Figure 2.6 Comparison between Wigner distribution and spectrogram (Hann window, length 32) for a sum of three periodic signals windowed in time, as shown in the left image. The components are better localised in time and frequency in the Wigner distribution, but there are cross-terms between the true components that obstruct the interpretation of the time-frequency distribution.

likely will not be expressible through a single component, cross-terms will often show up in Wigner distributions of these types of signals. Large parts of the field of time-frequency analysis deal with suppressing the cross-terms that arise in the Wigner distribution.

Hilbert transform For discrete data, the integral in (2.9) has to be replaced by a summation, which in itself contains a discrete Fourier transform (DFT). The discrete Fourier transform does not only introduce aliasing for frequency components between -0.25 and 0.25 (for frequencies normalised between -0.5 and 0.5), but also cross-terms between aliased terms and original terms. A nice way to solve this problem is by calculating the Wigner distribution for the Hilbert transform of the signal, also known as the *analytic signal* $z(n)$, defined as

$$z(n) = X(0) + 2\sum_{k=0}^{N/2-1} X(k)e^{i\frac{2\pi}{N}nk} \tag{2.12}$$

where X is the DFT of the original signal $x(n)$ and N is the signal length (assumed to be a power of 2) [Chan and Ho, 1990]. Essentially, the Hilbert transform removes the negative frequencies from the original signal, expanding the range of frequencies that can reliably be portrayed in the Wigner distribution to ± 0.5.

Ambiguity function

The ambiguity function is defined as the Fourier transform of the Wigner distribution, both in time and frequency. The frequency domain becomes a time-lag domain (τ domain) and the time domain becomes a frequency-lag domain (ν or Doppler-lag domain) after transformation. The ambiguity domain is therefore sometimes referred to as the (ν,τ) domain. Correspondingly there are four domains, but (t,f) and (ν,τ) are the ones that are most commonly used. The ambiguity function of the analytic signal $z(t)$ is defined as

$$A_z(\nu,\tau) = \int_{-\infty}^{\infty} z\big(t+\tfrac{\tau}{2}\big) z^*\big(t-\tfrac{\tau}{2}\big) e^{-i2\pi\nu t}\,\mathrm{d}t. \tag{2.13}$$

There are some well-grounded reasons for wanting to use the ambiguity domain. One is that, due to the two variables in the ambiguity function both being based on lag and not absolute times or frequencies, a signal can be time-shifted and frequency-shifted and remain unchanged in the ambiguity domain. Since the subjects in the study from which our data is gathered intrinsically have different response times to stimuli, a time-dependent feature extraction algorithm may have troubles simply because the event-related potentials in the subjects' EEG are occurring at slightly different times in each trial and between subjects. A similar case can be made for frequency variations between trials and subjects, but this is slightly more difficult to motivate.

Another reason for using the ambiguity function is that auto-terms tend to end up near the origin in the ambiguity function, while cross-terms tend to end up away from the origin after transformation. This can then be exploited by multiplying the ambiguity function with a function that has large values near the origin and small values otherwise, preferably where the cross-terms are expected to end up in the ambiguity domain. This type of function, which is a 2-D low-pass filter, is called an *ambiguity kernel*. The process of removing cross-terms using an ambiguity kernel is illustrated in Figure 2.7. Much of the research today is dedicated to developing kernels that keep essential parts of the Wigner distribution constant, e.g. the energy, while providing adequate cross-term suppression. A novel strategy is using the cross-terms for classification instead of the auto-terms, but most research is still aimed toward finding better ways of resolving the spectrum in the (t,f) domain and visualising it as well as possible.

Ambiguity kernels The filtered ambiguity function can be expressed as

$$A_z^F(\nu,\tau) = \int_{-\infty}^{\infty} z\big(t+\tfrac{\tau}{2}\big) z^*\big(t-\tfrac{\tau}{2}\big) \phi(\nu,\tau) e^{-i2\pi\nu t}\,\mathrm{d}t \tag{2.14}$$

where $\phi(\nu,\tau)$ is known as the ambiguity kernel. Applying the whole transformation back to the (t,f) domain, the filtered Wigner distribution $C(t,f)$ ends up being

$$C(t,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} z\big(u+\tfrac{\tau}{2}\big) z^*\big(u-\tfrac{\tau}{2}\big) \phi(\nu,\tau) e^{i2\pi(\nu t - f\tau - \nu u)}\,\mathrm{d}u\,\mathrm{d}\tau\,\mathrm{d}\nu \tag{2.15}$$

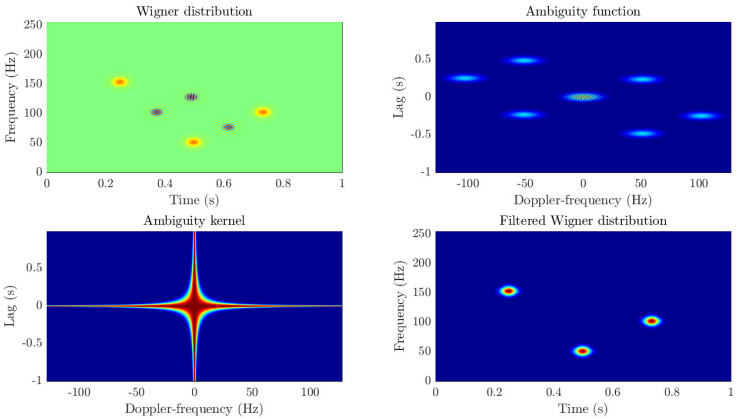

Figure 2.7 Illustration of the usefulness of ambiguity kernels. The plot in the upper left shows the same Wigner distribution as in Figure 2.6. The upper right image shows the same signal but in the ambiguity domain. The cross-terms can be seen as components away from the origin, in the middle of the figure. The plot in the lower left shows the Choi-Williams kernel [Choi and Williams, 1989], a common kernel function that works well at suppressing cross-terms between components simultaneously separated in both time and frequency. If two components appear at either the same time or the same frequency, this kernel will not work as well. The final image illustrates the resulting time-frequency distribution after filtering. The cross-terms are gone and we are left with an image that retains the well-resolved components and eliminates the unwanted cross-terms.

and $C(t,f)$ is said to belong to *Cohen's class* (or the quadratic, or bilinear class) of time-frequency distributions [Cohen, 1989]. If $\Phi(t,f)$ is the transform of $\phi(\nu,\tau)$ and $\underset{t\,f}{**}$ is the two-dimensional convolution in both time and frequency, then $C(t,f)$ can be rewritten as

$$C(t,f) = W_z(t,f) \underset{t\,f}{**} \Phi(t,f), \tag{2.16}$$

illustrating that $\Phi(t,f)$ can be interpreted as a smoothing filter for the Wigner distribution of the analytic signal $z(t)$. By analysing the ambiguity function, it is possible to remove cross-terms in a systematic fashion by designing an appropriate kernel. Discussions of desirable kernel properties can be found throughout literature [Jeong and Williams, 1992].

The spectrogram reinterpreted If the *time-lag kernel* $\rho(t,\tau)$ is defined as

$$\rho(t,\tau) = \int_{-\infty}^{\infty} \phi(\nu,\tau) e^{i2\pi\nu t}\,\mathrm{d}\nu, \tag{2.17}$$

then (2.15) can be rewritten as

$$C(t,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} z\big(u+\tfrac{\tau}{2}\big)z^*\big(u-\tfrac{\tau}{2}\big)\rho(t-u,\tau)e^{-i2\pi f\tau}\,\mathrm{d}u\,\mathrm{d}\tau. \tag{2.18}$$

The definition of the spectrogram from (2.3) can in turn be rewritten as

$$\begin{aligned} S_z(t,f) &= \left|\int_{-\infty}^{\infty} z(t_1)h^*(t_1-t)e^{-i2\pi f t_1}\,\mathrm{d}t_1\right|^2 \\ &= \left(\int_{-\infty}^{\infty} z(t_1)h^*(t_1-t)e^{-i2\pi f t_1}\,\mathrm{d}t_1\right)\left(\int_{-\infty}^{\infty} z(t_2)h^*(t_1-t)e^{-i2\pi f t_2}\,\mathrm{d}t_2\right) \\ &= \big[t_1 = u+\tfrac{\tau}{2};\ t_2 = u-\tfrac{\tau}{2}\big] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} z\big(u+\tfrac{\tau}{2}\big)z^*\big(u-\tfrac{\tau}{2}\big)h^*\big(u+\tfrac{\tau}{2}-t\big)h\big(u-\tfrac{\tau}{2}-t\big)e^{-i2\pi f\tau}\,\mathrm{d}u\,\mathrm{d}\tau \end{aligned} \tag{2.19}$$

and it is simple to identify

$$\rho(t,\tau) = h^*\big(-t+\tfrac{\tau}{2}\big)h\big(-t-\tfrac{\tau}{2}\big) \tag{2.20}$$

as the ambiguity kernel resulting in the spectrogram. This means that the spectrogram belongs to Cohen's class of time-frequency distributions, as well as the fact that the window function chosen in the spectrogram can both be analysed as a smoothing kernel and, in the same vein as before, designed cleverly to suppress cross-terms. In the case of the window h being a Hann window [Oppenheim and Schafer, 2009] with a specific length, the resulting kernel is very narrow and origin-centric, effectively resulting in low resolution in frequency and time but virtually no cross-terms.

Multitaper spectrograms When doing conventional time-frequency analysis, a common problem is that realisations (or trials) are very noisy—especially EEG—and thus the spectrum estimate may vary greatly between realisations. A method for tackling this is *multitaper spectral estimation*, which is done by estimating multiple spectra using statistically independent window functions. By averaging the spectra at the end, multiple independent estimates are obtained from only one realisation, so as to tackle the high variance problem with a relatively low price in bias [Xu et al., 1999]. Some common tapers used in multitaper estimation are the Slepian sequences (or discrete prolate spherical sequences) $|v_n^{(k)}(N,W)|$ which are defined, for each $k = 0,1,\ldots,N-1$, as the solutions to the system of equations

$$\sum_{m=0}^{N-1} \frac{\sin\big(2\pi W(n-m)\big)}{\pi(n-m)} v_m^{(k)}(N,W) = \lambda_k(N,W)v_m^{(k)}(N,W) \tag{2.21}$$

for $n = 0,\pm1,\pm2,\ldots$ where W is the chosen bandwidth and N is the number of windows [Slepian, 1978].

2.2 Common spatial patterns

The analysis of common spatial patterns (CSP) is another very common feature extraction method within EEG classification. While transform-based methods mostly deal with extracting as much information as possible from the EEG signals themselves, CSP takes on the approach of differentiating the signals based on similarities between signals across different channels. This has become one of the most common statistical feature extraction methods specifically for BCI-related research, while the community that focuses on anomaly and illness detection typically sticks to transform-based methods, even though the paper introducing the CSP method was applied to detecting neurological patients [J. Koles et al., 1990]. An advantage of CSP is that it finds similarities between channels, but the main disadvantage is that all temporal information is sacrificed in order to do so.

Essentially, the CSP algorithm does the following:

1. Estimate the covariance between all channels for each trial.
2. Create one covariance matrix for each class using the training data.
3. Simultaneously diagonalise (exactly for the two-class case and approximately for more than two classes) the class covariance matrices.
4. Use the eigenvectors corresponding to the largest or smallest ratio of eigenvalues between the covariance matrices as features for that class.
5. Classify validation/test data based on these features using some machine learning algorithm.

Each of these steps can be subtly changed and optimised for the problem at hand.

The two-class CSP problem

The main working assumption in CSP feature extraction is that information on which type of image the subject saw in each trial is encoded in variance changes of the EEG, and that, for each subject and class, the information-bearing components of the EEG originate in the same brain regions across trials [Grosse-Wentrup, 2008]. In essence, it is assumed that variance-based feature extraction can classify the EEG data.

Assume there are two classes, c_1 and c_2, to be distinguished in the data $\mathbf{x}$, and that classification is limited to one subject for now. Since it is assumed that each class has a specific covariance pattern, let $\mathbf{R}_{\mathbf{x}|c_i}$ be the covariance matrix of the class c_i. The *composite* covariance matrix is introduced as

$$\mathbf{R}_c = \mathbf{R}_{\mathbf{x}|c_1} + \mathbf{R}_{\mathbf{x}|c_2} \tag{2.22}$$

which can in turn be orthogonally diagonalised by

$$\mathbf{R}_c = \mathbf{P}_c \mathbf{\Lambda} \mathbf{P}_c^T \tag{2.23}$$

since the covariance matrix is symmetric. Now introduce the whitening transformation

$$\mathbf{W} = \mathbf{\Lambda}^{-1/2}\mathbf{P}_c^T \tag{2.24}$$

which whitens $\mathbf{R}_c$ by

$$\mathbf{W}\mathbf{R}_c\mathbf{W}^T = \mathbf{I}. \tag{2.25}$$

The point of this is that the transformed matrices $\mathbf{S}_{\mathbf{x}|c_1} = \mathbf{W}\mathbf{R}_{\mathbf{x}|c_1}\mathbf{W}^T$ and $\mathbf{S}_{\mathbf{x}|c_2} = \mathbf{W}\mathbf{R}_{\mathbf{x}|c_2}\mathbf{W}^T$ share the same eigenvectors, since $\mathbf{S}_{\mathbf{x}|c_1} + \mathbf{S}_{\mathbf{x}|c_2} = \mathbf{W}\mathbf{R}_c\mathbf{W} = \mathbf{I}$ [Fukunaga, 1990], i.e. if $\mathbf{S}_{\mathbf{x}|c_1}$ is decomposed by

$$\mathbf{S}_{\mathbf{x}|c_1} = \mathbf{U}\mathbf{\Psi}_{\mathbf{x}|c_1}\mathbf{U}^T, \tag{2.26}$$

then

$$\mathbf{S}_{\mathbf{x}|c_2} = \mathbf{U}\mathbf{\Psi}_{\mathbf{x}|c_2}\mathbf{U}^T, \tag{2.27}$$

where

$$\mathbf{\Psi}_{\mathbf{x}|c_1} + \mathbf{\Psi}_{\mathbf{x}|c_2} = \mathbf{I}. \tag{2.28}$$

Not only does this hold, but the eigenvalues to $\mathbf{S}_{\mathbf{x}|c_1}$ on the diagonal of $\mathbf{\Psi}_{\mathbf{x}|c_1}$ are in falling order such that

$$\mathbf{\Psi}_{\mathbf{x}|c_1,1} > \mathbf{\Psi}_{\mathbf{x}|c_1,2} > \cdots > \mathbf{\Psi}_{\mathbf{x}|c_1,P} \tag{2.29}$$

and consequently

$$\mathbf{\Psi}_{\mathbf{x}|c_2,1} < \mathbf{\Psi}_{\mathbf{x}|c_2,2} < \cdots < \mathbf{\Psi}_{\mathbf{x}|c_2,P}. \tag{2.30}$$

The purpose of performing feature extraction using CSP is to find the eigenvectors with the largest or smallest ratios between eigenvalues, because they will discriminate maximally between the two classes in the variance sense. The feature extraction step itself is done by extracting the m first and last columns of $\mathbf{U}$ (into, say, $\hat{\mathbf{U}}$) and projecting the data matrix $\mathbf{X} \in \mathbb{R}^{P\times N}$ (where P is the number of channels and N the number of samples in each trial) such that the features $\hat{\mathbf{X}} \in \mathbb{R}^{2m\times N}$ will be [Müller-Gerking et al., 1999]

$$\hat{\mathbf{X}} = \hat{\mathbf{U}}^T\mathbf{W}\mathbf{X}. \tag{2.31}$$

Interestingly, the aim of the authors describing this algorithm is also to classify into three classes, but they use a one against one scheme to do so. It has been shown that this type of scheme can yield worse results than direct multiclass CSP [Grosse-Wentrup and Buss, 2008], but this will be covered in the section below on CSP with more than two classes.

Covariance matrix estimation

In the previous part, it was assumed that there is a covariance matrix $\mathbf{R}_{\mathbf{x}|c_i}$ for the i^{th} class. These matrices need to be estimated from data. If the data matrix is $\mathbf{X} \in \mathbb{R}^{P \times N}$ for some trial, the covariance matrix is

$$\mathbf{R} = \mathbb{E}\left[\mathbf{X}\mathbf{X}^T\right] \tag{2.32}$$

where $\mathbb{E}[\cdot]$ denotes the expected value. Naturally, the expected covariance matrix is unknown, but it can be estimated by

$$\hat{\mathbf{R}} = \mathbf{X}\mathbf{X}^T \tag{2.33}$$

for a specific trial. In the original paper [J. Koles et al., 1990] it is suggested that normalisation of the covariance matrix with its trace is needed, but in more recent studies [Grosse-Wentrup and Buss, 2008; Blankertz et al., 2008] it is not done. Furthermore, it is suggested that trace normalisation can have a negative impact on pattern ordering [Larson et al., 2016].

However, the covariance matrix estimation is not finished there. The covariance matrix estimates need to be combined to form i "mean" estimates $\hat{\mathbf{R}}_{\mathbf{x}|c_i}$. If there are M_i trials for one given subject and class i, the estimate suggested in the original CSP paper is simply the mean covariance matrix

$$\hat{\mathbf{R}}_{\mathbf{x}|c_i} = \frac{1}{M_i} \sum_{k=1}^{M_i} \hat{\mathbf{R}}_{\mathbf{x}|c_i,k} \tag{2.34}$$

where $\hat{\mathbf{R}}_{\mathbf{x}|c_i,k}$ is the estimated covariance matrix of the k^{th} trial of class i. The MNE-Python toolbox [Larson et al., 2016] defaults to another type of "mean" covariance matrix estimation, where the M_i trials $\mathbf{X}_{i,k}$ are horizontally concatenated into a large data matrix $\underline{\mathbf{X}}_i \in \mathbb{R}^{P \times NM_i}$ from which the covariance is estimated using the standard formula (Eq. (2.33)). In our tests this combined covariance estimate proved to yield better results, so this estimation technique is used if nothing else is stated.

The combining of the covariance matrices from the test data is largely glossed over in literature, but a group of researchers is developing a toolbox for estimation using Riemannian geometry [Barachant et al., 2010] and a handful of other methods have been suggested. A number of covariance averaging techniques have been compared [Yger et al., 2015], and it was found that the Riemannian metric slightly outperforms the Euclidean metric for low channel numbers, while the Euclidean metric is best for large channel numbers (large being defined as 60 or more).

CSP with more than two classes

Since the original CSP algorithm depends heavily on eigenvectors on opposite sides of a matrix distinguishing between two classes, the extension to multiple classes is

not obvious. The problem of extending classification (and, in this case, feature extraction) is not entirely uncommon, however. Common schemes for circumventing the two-class limitation are for example one against one or one against all classification, see Section 3.1 on support vector machines. Schemes of these types make sense and are relatively easy to implement, but the complexity often grows quickly with the number of classes, and especially if large networks are trained, the computational cost can quickly become unbearable. Moreover, the final step of pooling the various classifications into one guess is often based on heuristics and is hard to support theoretically, motivating the search for algorithms that do multiclass feature extraction or classification in one step.

The first algorithm for direct multiclass CSP seems to have been presented in 2008 [Grosse-Wentrup and Buss, 2008]. It is based on identifying that the original CSP algorithm maximises an approximation of the mutual information between EEG components and classes, which in turn can be generalised to the multiclass problem. The concept of mutual information is well known in information theory, and appears once in a while in independent component analysis (ICA). It turns out that the algorithm suggested in that article, joint approximate diagonalisation (JAD) by maximising the same approximation of mutual information, is equivalent to ICA. The novelty in the article is that they provide a method for choosing the M independent components of greatest interest for discriminating between classes in order to minimise the classification error. Whether this approach gives better results or not is unclear, but it should definitely reduce computational complexity and simplify interpretation (being one set of spatial patterns instead of three for each combination of one against one or one against all).

3

Machine learning

Machine learning is a very broad field and we will limit this section to discussing some specific types of supervised learning that have been used during this project.

The goal of any supervised learning scheme is to find a model that maps a set of inputs to outputs in a way such that the model approximates the underlying function. To get an idea of how good this approximation is, we usually split the training data into two sets: training and test. The model generalises well if it scores well on the test set after training on the training set [Russell and Norvig, 2003].

3.1 Support Vector Machines

The support vector machine (SVM) is one of the most popular approaches for off-the-shelf supervised learning. One of the reasons why they are popular is that they usually generalise well even without domain specific knowledge. This is because they create a maximum margin separator, which means they find the boundary that has the largest possible distance to example points. It creates a linear hyperplane that separates the data, but using kernels the data can be embedded in a higher dimensional space that might not be linearly separable in the original space. This allows for flexibility to represent complex functions while still being resistant to overfitting.

One key insight of the SVMs is that they only consider the points closest to the boundary. This is clever since by choosing the margin furthest away from examples seen so far they attempt to optimise the generalisation loss rather than the empirical loss [Russell and Norvig, 2003].

Multiclass classification

SVMs can only classify between two classes but there are methods to extend this to many classes.

One against one This type of scheme uses one classifier for each pair of classes. When classifying, the class with the most votes wins.

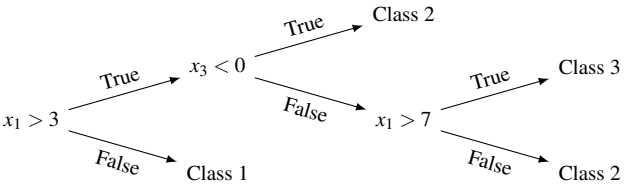

Figure 3.1 Example decision tree; x_i are inputs and the prediction of the class is the output.

One against all This scheme instead uses one classifier for each class. The data is split into the current class versus the rest for each classifier.

3.2 Decision trees

Training a decision tree (see Figure 3.1 for example) involves building the tree that maximises the information gain in each branching. Every deciding node will be based on a single parameter: the parameter that makes the children as "pure" (i.e. as low entropy) as possible [Russell and Norvig, 2003]. The decision tree is a very simple yet versatile algorithm that is often used in ensembles to turn its ability to overfit into an advantage. The errors for ensembles are based on subtracting the average errors made by the individual classifiers from the average variance of the classifiers, i.e. we want classifiers that are accurate but disagree with each other [Ohlsson, 2017]. Decision trees are good for this since we can have arbitrary accuracy depending on the depth we allow for the tree, and they vary considerably if they are trained on slightly different data.

Bagging This is an ensemble technique that creates new data sets by drawing samples from the original set. Data is drawn according to a uniformly random distribution with replacement to make learners train on different data sets.

Boosting This is an ensemble technique that weights the sample data points depending on how well they were classified by the previous learner. Thus, new learners will be more focused on data that previous classifiers did not manage to classify.

3.3 Linear discriminant analysis

Linear discriminant analysis (LDA) is a method of projecting the input space onto a lower dimensional space with good class-separability. It is similar to PCA, but where PCA looks for the component axis that maximises variance, LDA maximises component axis for class separation [Martínez and Kak, 2001]. Mathematically this is done by defining the:

1. *within*-class scatter matrix as

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (x_i^j - \mu_j)(x_i^j - \mu_j)^T \tag{3.1}$$

   where x_i^j is the i^{th} sample of class j, μ_j is the mean of class j, c is the number of classes and N_j the number of samples in class j, and the

2. *between*-class scatter matrix as

$$S_b = \sum_{j=1}^{c} (\mu_j - \mu)(\mu_j - \mu)^T \tag{3.2}$$

   where μ is the mean of all classes.

To separate the classes we want to maximise the between-class measure while minimising the within-class measure. One way to do this is to maximise the ratio $\frac{\det|S_b|}{\det|S_w|}$ [Martínez and Kak, 2001]. We used the implementation in scikit-learn [Pedregosa et al., 2011].

3.4 Artificial Neural Networks

Some of the early efforts to create artificial neural networks (ANNs) were inspired by the hypothesis that mental activity is nothing but electrochemical signals in a network of cells. This has later split into two disciplines: computational neuroscience that tries to model the brain to gain a better understanding of it, and a discipline that focuses more on the abstract and mathematical properties of ANNs that can be used in AI, such as the ability to process and learn from a very wide array of data [Russell and Norvig, 2003].

The basic component of any ANN is the neuron. A simple mathematical model for the neuron is $h_j = \phi(\sum_{i=0}^{n} w_{i,j} x_i + b)$ where h_j is the output of the neuron, ϕ is the activation function, x_i is the i^{th} input, $w_{i,j}$ is the weight of the connection from input i and b is a bias weight. The bias is usually incorporated into the network by adding a neuron with unit value in the previous layer (see Figure 3.2). The neurons are then connected in a network with some designated inputs and outputs and the signal is propagated from the input to the output by gradually calculating these sums and activation functions [Russell and Norvig, 2003].

There are two general types of artificial neural networks: feed forward networks (also know as multilayer perceptrons, MLPs) and recurrent networks. MLPs are built by layers of nodes and all data only travels in one direction layer by layer. Recurrent networks have loops in the connections and can thus depend on data from previous passes through the network. We will make use of both and give a more detailed introduction to the specific types we use later on.

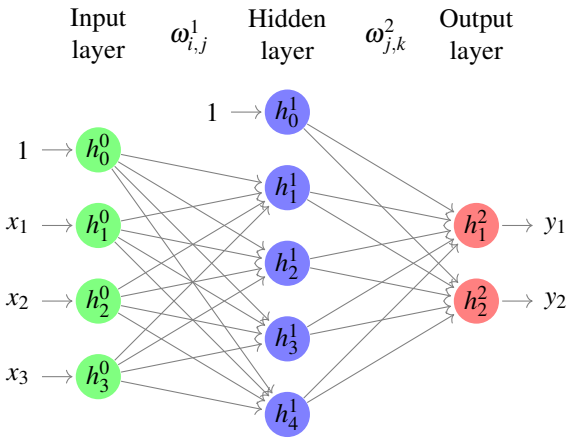

Figure 3.2 Neural network with one hidden layer. All layers except the output layer have a bias node with value 1.

Cost function

To be able to train a network we need to define what the target is and define a measure of how large the current error is. A cost function takes the predicted values y_k together with the target values d_k and produces a value for the current error.

The best cost function will vary depending on the type of problem, i.e. regression and classification problems will typically use different types of cost functions. The functions are commonly derived from the *maximum likelihood* principle [Ohlsson, 2017].

Mean Squared Error A cost function that is commonly used when working on regression problems is the mean squared error. The assumption is typically that the labels are measurements of the underlying function with some Gaussian noise added. The whole cost function E is

$$E(y) = \frac{1}{2}\sum_{k=1}^{n}(y_k - d_k)^2 \tag{3.3}$$

Categorical crossentropy This is a cost function that is commonly used when working on classification problems where we interpret the output as class probabilities. It works for 1-out-of-C coding on classification problems with C classes. It is most often used with the *softmax* activation function after the last layer to make the output class probabilities (see Eq. (3.8)) [Ohlsson, 2017; Chollet et al., 2015].

$$E(y) = -\sum_{k=1}^{n} d_k \log y_k \tag{3.4}$$

Gradient based learning

Training an ANN with gradient descent is very similar to training other models with the exception that the non-linearities in the activation function cause the cost function to become non-convex. Both linear and logistic regression as well as SVMs have global convergence—a guarantee we do not have for ANNs. Instead of using a linear equation solver to find the minimum we rather employ an iterative approach where the weights are updated in the direction of the negative gradient. This will find a minimum of the cost function, though it might not be the global minimum [Goodfellow et al., 2016]. The vanilla gradient update equation is

$$\omega_{j,k} \leftarrow \omega_{j,k} - \eta \cdot \frac{\partial E}{\partial \omega_{j,k}} \tag{3.5}$$

where η is the learning rate.

Backpropagation

Updating the weights in the last layer seems easy enough since the cost is defined for actual labels and predicted values, both of which we have. How do we then update weights in the middle of the network where there are no pre-defined labels?

The answer is to use backpropagation. The algorithm works by saving the values for every node when the forward pass is done. Then, starting from the last layer, the cost is calculated and propagated backwards through the net. All intermediate updates can be calculated with the chain rule following Equation (3.5). The whole process looks as follows [Ohlsson, 2017]:

1. Choose an input μ and set
$$h_k^0 = x_k(\mu), \quad \forall k$$
2. Propagate the signal forward in the network by
$$h_i^m = \phi_i^m \left(\sum_j \omega_{ji}^m h_j^{m-1} \right), \quad \forall i, m$$
3. Calculate "deltas" for the output layer M
$$\delta_i^M = \phi' \left(\sum_j \omega_{ji}^M h_j^{M-1} \right) \left(d_i(\mu) - h_i^M \right)$$
4. Calculate "deltas" for the remaining layers by propagating the error backwards
$$\delta_i^{m-1} = \phi' \left(\sum_j \omega_{ji}^{m-1} h_j^{m-2} \right) \sum_j \omega_{ij}^m \delta_j^m$$

for $m = M, M-1, \ldots, 2$

5. Update the weights

$$\Delta\omega_{ij}^{m} = \eta\,\delta_i^m h_j^{m-1}$$
$$\omega_{ij}^{m}(t+1) = \omega_{ij}^{m}(t) + \Delta\omega_{ij}^{m}$$

6. Repeat for next input.

One problem that arises for certain types of networks when trained with back-propagation is that the gradient vanishes exponentially with the depth of the network [Hochreiter and Schmidhuber, 1997; Bengio et al., 1994]. This happens since there is a chain of multiplying derivatives of activation functions when the error is propagated up through the layers. The absolute value of derivatives of activation functions is commonly less than one, resulting in the vanishing gradient. If the size of the gradient is instead larger than one, the opposite problem with exploding gradients will arise.

A vanishing gradient is common for both convolutional networks, that usually depend on many layers of encoding features, and recurrent networks, that can be very deep when unfolded if there are many time-steps in the sequence. There are methods for avoiding this problem and some specific solutions for the different networks will be presented in the coming sections.

Activation functions

Activation functions are needed to introduce a non-linearity in the network. Without them the network is only a linear classifier no matter how deep it is made [Russell and Norvig, 2003]. It has been proven that feed forward networks, with a single hidden layer and some weak conditions on the activation function, are universal approximators if there are enough nodes [Hornik, 1991].

Rectified linear unit (ReLU) is one of the standard activation functions for deep learning. The ReLU activation function is defined as

$$\mathrm{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3.6}$$

One of the main advantages of ReLU is that it avoids the vanishing gradient problem by having its derivative be 1 for any positive input. This will however shift the mean activation to be positive, since any output from the function is either positive or zero. A shift in mean activation can pose some problems; if a unit has a non-zero mean activation it will act as a bias towards the next layer. If the average over such units in a layer does not cancel out, training will cause a bias shift for units in the next layer. Less bias shift will cause the learning to speed up [Clevert et al., 2015] and therefore we want to bring the mean activation closer to zero.

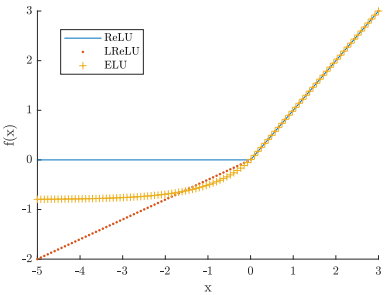

Figure 3.3 ReLU, LReLU and ELU activation functions.

Leaky ReLU and Parametric ReLU A proposed fix for the non-zero mean activation is to replace the negative part with a linear function, know as Leaky ReLU (LReLU). They have been shown superior to ReLUs in both learning speed and performance [Maas et al., 2013]. A generalized version, Parametric ReLU (PReLU), where the slope is a trainable parameter, has later been suggested [He et al., 2015]. PReLUs have shown good performance on image data sets where the learning behaviour is better than with ReLUs [He et al., 2015].

Exponential Linear Unit LReLU and PReLU suffer from a different problem though, they no longer ensure a noise-robust deactivation state [Clevert et al., 2015]. The same study also suggests a new activation function, Exponential Linear Units (ELU), with negative values to allow a mean activation close to zero, but at the same time saturate to a negative value with decreasing input. This model will encode the presence of a phenomenon and to what degree, but not quantitatively model the absence of it. This results in a model with a noise-robust deactivation state.

ELU is defined for $\alpha > 0$ as

$$\mathrm{ELU}(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases} \tag{3.7}$$

See Figure 3.3 for comparison of the different functions.

Softmax We also have the activation function *softmax* that is used in multiclass classification. It is most often used in the last layer to ensure the output probabilities y_k are in the range $[0, 1]$ and fulfil $\sum_k y_k = 1$ [Ohlsson, 2017].

$$softmax(\mathbf{x})_i = \frac{e^{x_i}}{\sum_k e^{x_k}} \tag{3.8}$$

Batch normalisation

It has long been known that normalising input to zero mean and unit variance reduces training times [LeCun et al., 1998]. Intuitively, if there were no extreme outliers in the input space, the weights would not have to vary as much to minimise the

error, and the distance between the random initialisation and the minimum would on average be closer.

Following this there is a technique known as batch normalisation [Ioffe and Szegedy, 2015]. The idea is that since the output of every layer acts as input for the next layer, we normalise between every layer. This is not done over all weights at the same time, but rather over every mini-batch that the network is trained on.

Passing a mini-batch $B = x_1..x_m$ of input values x_i through the algorithm produces an output y_i according to:

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$

where γ and β are trainable parameters and ε is a small constant used for numerical stability.

Inserting this directly into state of the art models for image recognition shows improved accuracy and training times. Batch normalisation could also allow for other modifications such as removing dropout layers and increasing learning rates with little to no loss in accuracy [Ioffe and Szegedy, 2015]. This would further improve training times.

Data augmentation

One straightforward way of reducing overfitting and improving classification in most machine learning applications is to get more data to train on. This may not always be easy, as for example in our case. Data augmentation is a technique successfully used on image data sets with a low number of samples [Krizhevsky et al., 2012]. The images are artificially altered in a label preserving way to enlarge the data set.

For images this can entail zooming, skewing, translating, mirroring, adding noise, changing color intensities and many other techniques.

For EEG data it is slightly harder to know what is label preserving, but we reason that adding Gaussian noise should be one good option (the data is already full of noise) and using a small random offset in time could be a second option (slightly different reaction times among subjects over trials).

Long short term memory

Long short term memory (LSTM) networks, originally introduced in [Hochreiter and Schmidhuber, 1997], are a type of recurrent neural networks. The problem with

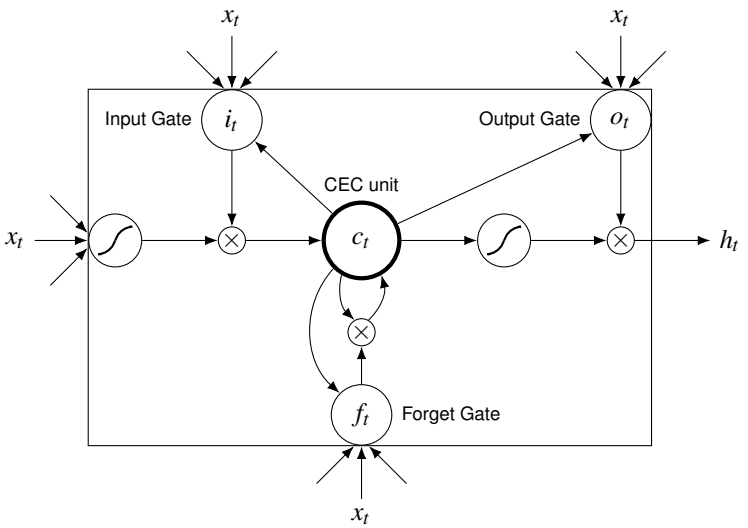

Figure 3.4 There are a few different versions of LSTM units. This is an example of a peephole unit where the gates have direct access to c_t.

simple recurrent networks is that the error signals propagating back through time tend to vanish from many derivatives of activation functions being multiplied together [Ohlsson, 2017]. The solution that is introduced with LSTMs is that the error signal is forced to stay constant by the architecture of the units and the training algorithm. Therefore LSTMs are good at bridging information over larger times without losing any short time lag capabilities.

The internals of an LSTM unit revolve around a self-connected linear unit (known as the constant error carousel, CEC) with gates that control information flow to and from it. The *input gate unit* is there to protect the memory contents from irrelevant noise. Similarly the *output gate unit* is there to protect the output from memory content that is irrelevant at the moment. There is also a *forget gate unit* that controls how much of the memory stays [Gers and Schmidhuber, 2001]. Figure 3.4 show a sample of a single peephole unit, this is where the gates have direct access to the CEC.

Convolutional neural networks

Convolutional neural networks (CNNs) typically consist of several convolutional, pooling and dropout layers with a few fully connected layers at the end. They are very good when there is local dependence in the data which is used to compute higher order features in following layers. They also have far fewer weights than fully connected models with the same number of hidden neurons which will make them easier to train while still allowing for complex functions.

A common way to visualise what is happening in convolutional networks is through studying the filters. The higher values in the filter encode the spatial arrangement of important features and when plotted as an image the features are easily visualised [Goodfellow et al., 2016].

Convolutional layers Convolutional layers are the core idea behind CNNs. They consist of filters sliding over the input to extract local features for use in the next layer. This results in some important properties such as *sparse interactions*, *parameter sharing* and *equivariant representations* [Goodfellow et al., 2016].

Sparse interactions mean that every node only interacts with a few of the nodes in the input. This allows for fewer parameters in the network and improves both statistical efficiency and memory requirements.

Parameter sharing further reduces the storage size of the parameters by having many nodes share the same weights.

Convolutions are equivariant to translations, but not necessarily to all transformations.

For example, if we have 31 channels of 512 time instances we might have 10 1D filters of width 64. However, every filter is a combination of the current window over all channels at the same time. Thus there will be 10 filters of size 31×64 creating 10 "channels" of output with 512 time instances each. On the contrary, if we wanted the same input and output sizes for a fully connected network we would have around $31 \cdot 512 \cdot 10 \cdot 512 \approx 8 \cdot 10^7$ parameters, which is a few orders of magnitude more.

Pooling layers Pooling is a way of summarising the output of neighbouring groups in the same filter map. This subsampling will make the network less sensitive to small displacements of features, since positions of features are "blurred" when the subsampling is done. This also helps with reducing the size of the data and allowing more feature filters while still keeping the number of weights down [Goodfellow et al., 2016].

For example, if we have 31 channels of 512 time instances we might have a filter of size 4 which scans over each channel and results in $v_i^k = \max(q_{4i..4i+3}^k)$. Here k is the channel, $0 \leq i < 128$ is the window position, q the original data of size 31×512 and v the subsampled data of size 31×128.

Dropout This technique was introduced to improve generalisation error in large networks [Hinton et al., 2012]. It can be seen as an effective regularizer by forcing the network to not use brittle co-adaptations of hidden units. It works by randomly shutting off units with a probability r during training, while letting all units be part of prediction. Since this will result in fewer weights feeding into a unit the value is scaled by $\frac{1}{1-r}$ during training. This can also be seen as randomly training an ensemble of networks with a different number of units in the hidden layers. This will allow us to use larger models without overfitting, at the cost of longer training times [Dahl et al., 2013].

The feedforward equation during training will be

$$h_j^m = \phi\left(\frac{1}{1-r}\sum_i h_i^{m-1} q_i^{m-1} \omega_{ij}^m\right) \tag{3.9}$$

where $\boldsymbol{q}^m$ is a binary mask for layer m with entries drawn from $Bernoulli(1-r)$.

Fully connected layer After the convolutional layers there is often a fully connected layer (or a few) to provide the final classification from the features that have been extracted.

4

Methods

4.1 Data set and pre-processing

The EEG data used in this work is from a study performed at Department of Psychology in Lund by Mikael Johansson and Inês Bramão [Bramão and Johansson, Submitted for publication, 2018]. The study was conducted in a Faraday cage using a Neuroscan Grael amplifier from 31 Ag/AgCl scalp electrodes.

There were 36 subjects, both female and male, divided equally in two groups of 18 for the visual and verbal task. They were all native Swedish speakers without any known neurological or psychiatric disorders. Subjects were asked to look at pictures from three classes (*encoding* or *study* phase) and later remember information about the pictures (*retrieval* or *test* phase), see Figure 4.1. The classes were faces of famous people, famous landmarks such as the Niagara Falls and man-made objects such as a violin or a hammer.

The *study* phase was the same for both tasks. The subjects were looking at a screen where they were shown a word for 2.5 s, an image for 2.5 s, and both together for 2.5 s. The 2.5 s of only seeing the image was the data used for training.

The *test* phase presents the word from study, after which the subject responds with the class of the corresponding picture. If the subject is correct on the class the visual task presents the image together with a mirrored version while the verbal task asks for the name of the person/place/object in the image. If the class was wrong, no such task was given and the test proceeded. The data used for classification is the word presented in the start of the test phase.

The data was pre-processed in FieldTrip, a MATLAB toolbox for EEG and MEG analysis. It was downsampled from 2048 Hz to 512 Hz and divided into epochs of 4 s, -1.5 s to 2.5 s relative to the onset of the image in *study* and the onset of the word in *test*.

Eye movement and other muscular activity can introduce artefacts in the data and needs to be analysed and removed. The EEG data was physically inspected for artefacts from muscular activity or other activity not related to eye movement which were manually removed. Artefacts from eye movements were removed by

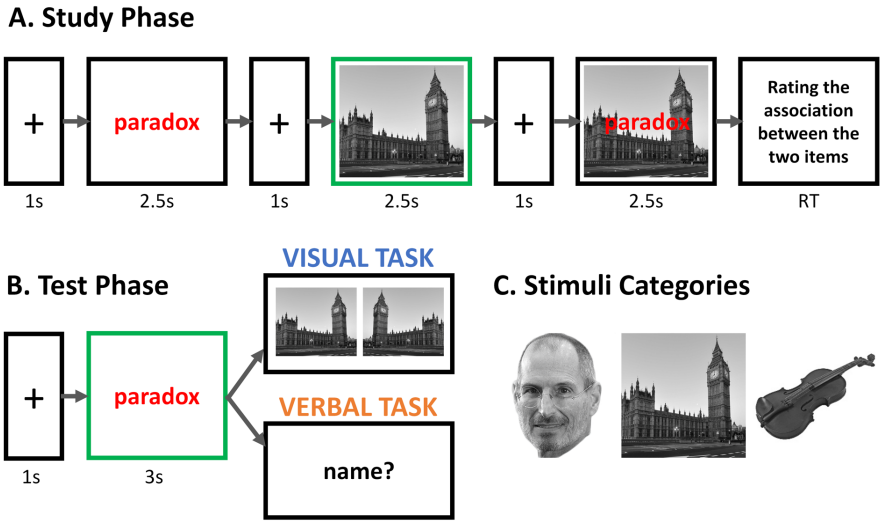

Figure 4.1 (A) shows the encoding phase where subjects looked at word and picture pairs. This phase is identical for both the visual and verbal task and the EEG signals from when the picture was presented alone (outlined in green) is the data used for training. (B) shows the retrieval phase. For both tasks the subjects are first asked to retrieve the category of the picture from the word. The visual task was then to determine which of two mirrored images was the original one. The verbal task was then to name what was in the picture. The data used for classification was the EEG signals from the word being presented, also outlined in green. (C) shows examples of images from the different classes (faces, landmarks and objects). The image is borrowed from [Bramão and Johansson, Submitted for publication, 2018] with the authors' permission.

using independent component analysis. Trials were also visually inspected and any trials with residual artefacts were removed.

4.2 Experimental setup

For the machine learning we used GPU nodes on LUNARC, a scientific computing cluster in Lund. This yielded an increase in speed of around 10 times compared to running on the CPU nodes when testing with some deeper networks, while for smaller networks it could run slower.

We ran TensorFlow [Martín Abadi et al., 2015] with Keras [Chollet et al., 2015] on top for the neural networks. For other machine learning approaches we used implementations from scikit-learn [Pedregosa et al., 2011] and MATLAB.

For feature extraction/transformations we used mne-python [Larson et al., 2016] and MATLAB—both existing tools in MATLAB and some tools that were written

during the project.

We used the data from the 18 subjects in the visual task, both encoding and retrieval, for all the accuracies presented in results. To estimate generalisation performance of the models we used 10-fold cross validation. This is a method to estimate generalisation performance over all the data by splitting the data in ten parts. Over ten iterations training is done on nine of them while testing is done on the remaining one, and the generalisation performance is approximated to be the average of the ten values of the classification accuracy. This is what is later called *encoding accuracy*. We also present results for *retrieval accuracy* which is a model trained on data from the encoding phase and tested on data from the retrieval phase.

When using any measure of the performance of the models we always run the risk of finding a model that fits extra well to the current data under this measure. This is also true for cross-validation, and over many iterations of improving a model we will most likely introduce some type of bias towards this measure into the structure of the model. To test if this was the case for us we only ran the models on the visual encoding data while improving them, then in the end we ran the model on the verbal encoding data with similar results. From the results of this test we concluded that we did not have any significant bias in the model selection and that our estimate of the generalisation performance of the model was still good.

Simple classifiers

We ran a few simple classifiers on the different transformations of the data that we generated. We did not change the parameters for the classifiers significantly; instead we were interested in using these initial training experiments as a quick search for which transforms contained good information and thus were worth testing with more advanced classifiers. Before classification, no matter what transform has been done, every trial is stacked to be a 1D vector. A trial is what corresponds to 31 channels of 2049 time instances in the original data. The input data was cut from 0 s to 1 s relative to onset if not otherwise stated. This was done to reduce the dimension of the input space by removing data that we assumed did not contain much information. Additionally, if decimation is mentioned, the data is low-pass filtered and downsampled using the `decimate` function in MATLAB. This operation also serves to reduce the input space by removing the highest frequencies, which we assume to contain little or no information useful for classification.

The SVMs used have the basic implementation with a linear kernel from MATLAB and scikit-learn. We tried using regularisation but did not see much of a difference other than in computation time, so the data in the results is from classifiers without regularisation.

The LDAs used have the basic implementation from MATLAB and scikit-learn [Pedregosa et al., 2011].

The decision trees use the basic implementation from MATLAB. We also test ensembles of trees with either bagging or boosting to increase the model complexity.

Table 4.1 Design of the LSTM net that generalised the best. We tried varying the parameters and structure of the network. Return sequences is a setting in Keras that switches between only returning the last value or the entire predicted sequence. This network has approximately 70000 trainable parameters.

Layer	Nodes	Additional settings
LSTM	70	return sequences
Dropout	—	0.4
LSTM	70	—
Dropout	—	0.3
Dense	15	tanh
Dense	3	softmax

Long Short Term Memory

The LSTM network was our first idea for a neural network to try. They are well known and well used for time series data which seemed like a perfect fit. See Table 4.1 for the layout of the net. The input data was cut in time from 0 s to 1.5 s (768 time instances in total). This is intentionally slightly longer than what was used in the simple classfiers, partly because these are a bit more complex and can probably better handle more data. The other reason is that we ran some tests where we got lower accuracy for both the full four seconds of data as well as only one second of data. Therefore we went with 1.5 seconds as the standard data length for the neural networks and CSP algorithms.

1D convolutional network

The 1D convolutional networks (CNN1D) were our second idea; see Table 4.2 for the layout of the net. The thought was that we might not want the full history of the temporal data in the way that LSTM can encode, but rather just have a dependence on the nearest neighbours in time. The networks were fine tuned by testing and gathering inspiration from papers on similar problems. The input data was cut in time from 0 s to 1.5 s (768 time instances in total). It was also normalised over trials and time for each subject separately (average and variance over channels are 0 and 1 for every subject, respectively).

The two first layers are for *data augmentation*; the first creates a random offset in the data while the second adds Gaussian noise.

When decimation is used, the data is low-pass filtered and downsampled in scipy's `decimate` function.

EEGNet

The EEGNet was implemented according to the original EEGNet paper [Lawhern et al., 2016]. This net was also fed with data that was cut from 0 s to 1.5 s. This network has around 5000 trainable parameters.

Table 4.2 Design of the 1D convolutional network we used. This is the "base" CNN1D network, and in our results we compare this to similar networks were we vary parameters. This network has approximately 90000 trainable parameters. The layer names correspond to existing layers in the Keras library, except for the offset layer which we wrote ourselves.

Layer	Nodes	Additional settings
Offset		slice 630 randomly from 768
GaussianNoise		~0.01
Conv1D	30	filter size 64, causal padding
BatchNormalisation		
Activation		ELU
AveragePool1D		pool size 2
Dropout		0.2
Conv1D	15	filter size 32, causal padding
BatchNormalisation		
Activation		ELU
AveragePool1D		pool size 2
Dropout		0.3
Conv1D	10	filter size 16, causal padding
BatchNormalisation		
Activation		ELU
AveragePool1D		pool size 2
Dropout		0.4
Flatten		
Dense	15	
BatchNormalisation		
Activation		tanh
Dense	3	softmax

Table 4.3 Design of the first 2D convolutional network net we ran on the transformations. The second version of the net (CNN2D—v2) had filters of size (8, 4) instead and had pooling of size (4, 4). The layer names correspond to existing layers in the Keras library.

Layer	Nodes	Additional settings
Conv2D	4	filter size (4, 8), same padding
Activation		ELU
AveragePool2D		pool size (2, 4)
Dropout		0.2
Conv2D	8	filter size (4, 8), same padding
Activation		ELU
AveragePool2D		pool size (2, 4)
Dropout		0.3
Conv2D	16	filter size (4, 8), same padding
Activation		ELU
AveragePool2D		pool size (2, 2)
Dropout		0.3
Flatten		
Dense	15	tanh, L1 reg=0.01
Dense	3	softmax

2D convolutional network

Using a 2D convolutional network (CNN2D) on time-frequency distributions is not really something we have seen other people do. There is some work done on spectrograms, and to our knowledge, one paper which tests it for the Wigner distribution [Brynolfsson and Sandsten, 2017]. We reason that there is local dependence in time and most likely also some local dependencies in frequency. The time-frequency transforms result in 31 channels of 2D matrices, very similar to RGB images that have multiple color channels of 2D matrices, i.e. we can treat them with the same or similar 2D convolution tools that are developed for images. We tried two versions of a network since the different transforms created matrices of varying shape. The first version is in Table 4.3 while the second version (CNN2D—v2) was the same except the filters were of size (8, 4) and pooling of size (4, 4).

Time-frequency transforms

For the time-frequency transformations there are a handful of parameters to consider. The data being transformed was cut from 0 s to 1.5 s (768 time instances) and decimated with a factor 4 with MATLAB's `decimate` function. The sampling

frequency parameter was thus chosen as $F_s = 128$ Hz. The window length, both for the single Hann window in the spectrogram and the Slepian windows in the multitaper spectrogram, were chosen to be $L = 32$ samples long, or 250 milliseconds. Some tests were done with $L = 8$ or 62.5 ms windows with lower accuracy. The number of FFT points was chosen as 512, which is the smallest power of two that surpasses double the signal length, $2 \cdot 768/4 = 384$ samples, or one and a half second of downsampled data after onset. The Slepian windows were calculated using `dpss` in MATLAB with `NW=3` and the last three windows discarded. In the Wigner and ambiguity tests, no special kernels were applied.

Common spatial patterns

The common spatial patterns algorithm was run on data in the interval 0 s to 1.5 s relative to onset. We tried varying a number of parameters. The default settings are concatenated covariance estimation, no trace normalisation, and four independent components. We tried the algorithm with trace normalisation, epoch covariance estimation, and also with eight components. See the part on covariance matrix estimation in Section 2.2 for more details.

Significance of results

It is tempting to simply assume that all classification errors are binomially distributed with $p = \frac{1}{3}$ and to set up a confidence interval based on this assumption. However, due to the 10-fold cross-validation scheme this assumption is not so obvious, and there are several articles indicating that this is a very bad idea [Refaeilzadeh et al., 2009; Lacoste et al., 2012; Bouckaert and Frank, 2004]. For any pair of training-validation procedures in a 10-fold cross-validation scheme, eight out of the ten folds will be used for training in both. This is the main argument for the classification errors not simply being independent and identically distributed over the whole cross-validation scheme.

There are multiple alternatives to the simple hypothesis test, many of which are described in a very thorough work published in 2003 [Nadeau and Bengio, 2003]. One of the schemes is based on bootstrap—by randomly permuting the class labels and retraining the network repeated times, we can obtain a sample from the distribution of the cross-validation accuracy. Then we can estimate an approximate confidence interval from this data and compare it to the cross-validation accuracy obtained from training the network on the correct class labels. Unfortunately, for a few of the slower neural networks such as EEGNet and CNN2D, we have not had time to do significance testing. These networks can take around 5–15 hours for running one 10-fold cross-validation, so running every one of them a sufficent amount for a good confidence interval would take more time than we had at hand.

5

Results and discussion

5.1 Raw data with simple classifiers

Table 5.1 The main test on raw data with simple classifiers. We tried other decimation factors, producing similar results. Overall, decimation does not change much for these classifiers, but some algorithms are definitely more successful than others.

Classifier	Data	Encoding acc.	Retrieval acc.
SVM	Raw cut	0.74	0.34
LDA	Raw cut	0.77	0.34
Tree	Raw cut	0.48	0.33
Trees with bagging	Raw cut	0.66	0.34
Trees with boosting	Raw cut	0.69	0.34
SVM	Raw cut, 16x deci.	0.75	0.34
LDA	Raw cut, 16x deci.	0.74	0.34
Tree	Raw cut, 16x deci.	0.52	0.34
Trees with bagging	Raw cut, 16x deci.	0.69	0.34
Trees with boosting	Raw cut, 16x deci.	0.71	0.34

The results in Table 5.1 are slightly surprising, since these accuracies are extremely high compared to how unsophisticated the method is. All of the classifiers perform similarly or slightly better after decimating the data by up to a factor of 16, indicating that there is a lot of redundancy in the high frequencies of the data.

5.2 Raw data with different neural networks

LSTMs gave decent results when we started out, but they did not generalise as well for every subject when we started running 10-fold cross-validation.

The CNN1D networks did much better which might show that the local patterns are more important than a full temporal dependence. We still get semi-deep net-

Table 5.2 Some of the more successful and interesting neural networks we ran on raw data. We tested how the components of the network affected performance by running the base CNN1D with and without them. See table 4.2 for base CNN1D implementation.

Classifier	Data	Encoding acc.	Retrieval acc.
LSTM	Raw cut	0.41	0.33
CNN1D	Raw cut	0.79	0.35
CNN1D	Raw full data	0.77	0.35
CNN1D, no data aug.	Raw cut	0.82	0.35
CNN1D, no batchnorm	Raw cut	0.76	0.34
CNN1D, PReLU	Raw cut	0.77	0.34
CNN1D, MaxPool	Raw cut	0.79	0.34
CNN1D, no data aug.	Raw cut, 4x deci.	0.79	0.35
CNN1D, no data aug.	Raw cut, 16x deci.	0.79	0.35
CNN1D, 3 spread channels	Raw cut	0.50	0.33
CNN1D, 3 top channels	Raw cut	0.63	0.34
CNN1D, 7 top channels	Raw cut	0.75	0.35
EEGNet	Raw cut	0.75	0.34

works to find complex features but with far fewer parameters to train. Based on the idea to interpret convolutional networks for images through their filters, we tried that on the CNN1D networks as seen in Figure 5.1. However, as the 1D-data is noisy and not very intuitive, the filters themselves were hard to interpret and we were not able to discern any specific features in them. This led us to instead average the weights in the filters of the first layer over all dimensions but the channels, see Figure 5.2. The idea is that larger weights in general means they encode more important features and larger weights in general means more important features in that channel. The high activity (and perceived importance) of the channels in the back of the head is similar to findings in previous work, but also very reasonable since the visual cortex is located there. When training a network on only three spread out channels (PO10, T7, Fp2) we get an accuracy of 50 %, while training on the top three channels (P8, PO9, PO10) gave an accuracy of 63 % (see Table 5.2). This seems to confirm that the channels in the back are more important, and when training on all the seven channels in the back (P7, P8, PO9, PO10, O1, O2, Iz) we got 75 % accuracy. So while that is good classification from only a quarter of the channels, there is still data with additional classifying power in the remaining channels.

Also worth mentioning is that EEGNet got an accuracy of 75 % without any tuning of the hyperparameters from our side. The EEGNet use many of the same techniques as we use, but structure the network slightly differently to reduce the number of weights.

Looking at Figure 5.3 we see that CNN1D on encoding data classify with high-

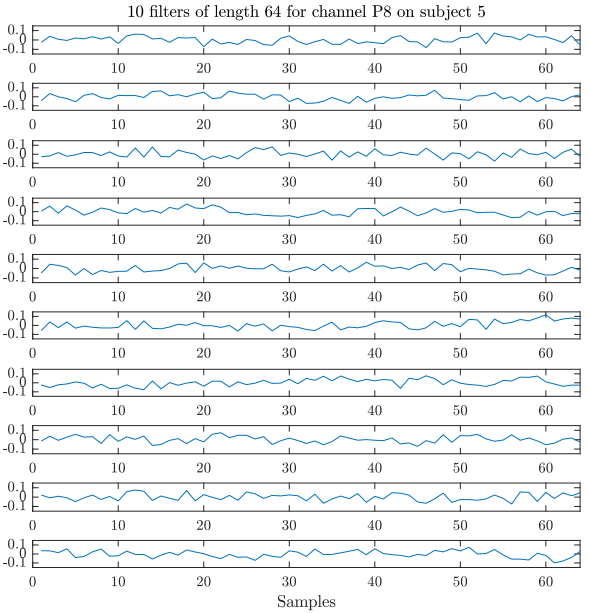

Figure 5.1 10 filters from the first convolutional layer of a CNN1D network trained on channel P8 for subject 5 in the visual task.

est accuracy around 200–300 milliseconds post stimulus onset. Comparing this to Figure 5.4, which shows the classification accuracy on retrieval data after training on encoding data in different time bins, we notice that the best classification is delayed. It seems like there is highest classification accuracy around 800–1000 milliseconds for both the training and classification bins, but as the histogram on the right shows, the classification accuracies overall are evenly distributed around the mean classification accuracy, 33.23 %. This indicates that, on average, the classification accuracies are no better than random guesses, even though some cases technically are. We believe that it would be inaccurate to make the claim that certain time intervals hold more classification power than others, although we have not conducted a formal hypothesis test. Figure 5.6 shows a similar test where the classification accuracies are similarly distributed but there is even less of a pattern in the time-dependent classification.

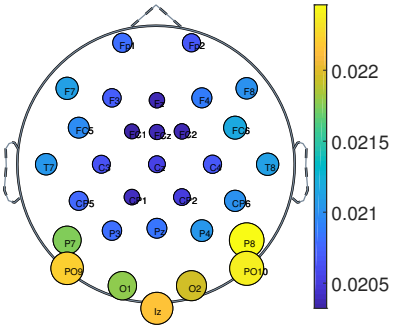

Figure 5.2 Average absolute filter-weights for every channel in the first layer of a CNN1D network. The weights are generated from 10 repeated runs on 18 subjects with 10-fold cross-validation, sum over kernels of length 64 for 30 filters. If we look at the distribution of how large the average absolute weights are for every channel, there are 14 in the lower quarter and 5 in the upper quarter. The CNN1D network trained on only the 3 channels with highest average weights got 63 % accuracy, see Table 5.2.

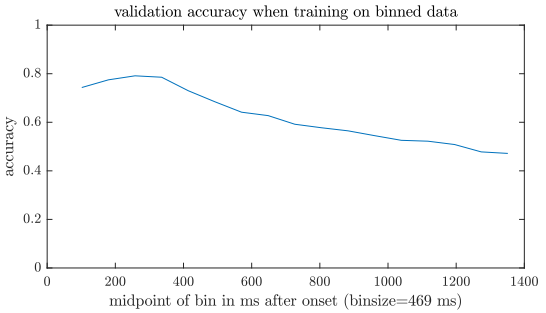

Figure 5.3 10-fold cross-validation accuracy for CNN1D networks on binned encoding data. Each classifier was trained on *bins* of 240 consecutive time instances (469 ms) with the distances between the midpoints of each *bin* being 60 time instances (117 ms). That means every classifier is trained on data that stretches from 234 ms before to 234 ms after the time on the x-axis.

5.3 Time-Frequency transforms with 2D convolutional networks

Neural networks can be shown to be universal approximators if enough nodes are used, i.e. any transform of the data should not be necessary for a neural net's ability to encode the data (no new information is added, just transformed). Even though

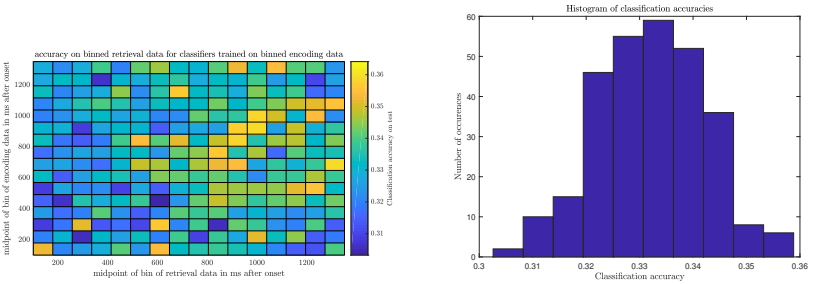

Figure 5.4 (Left) Classification accuracy for CNN1D networks trained on binned encoding data and tested on binned retrieval data. Each classifier was trained on bins of 240 consecutive time instances (469 ms) from the encoding data with the distances between the midpoints of each *bin* being 60 time instances (117 ms). Testing was done in the same fashion with *bins* of size 240 time instances and a distance of 60 time instances in between, but now using the retrieval data. (Right) Histogram over all the values in the matrix. The mean classification accuracy is 33.23 %.

they are theoretically not needed, transforms might be important for the simple reason that having too many nodes/parameters is not feasible in practice. We are only working with about 180 data points (trials) per subject so if we can have fewer parameters to train while still being able to encode the data it will help with for example reducing training times and avoiding overfitting.

The transformations with CNN2D networks seemed good but never fully reached the same accuracy as the 1D on the raw data, as we can see in Table 5.3. This is something we would have liked to spend more time on since we think the transformations might be able to give more interesting and more interpretable features than raw data. The one big problem is that the transforms are very big in size and take very long time to train on which makes them much more cumbersome to work with. On a broader scale, however, the method of 2D convolutional neural networks seems very promising since current classifying methods within the field are often based on simpler classifiers and further feature extraction methods for dimensionality reduction. 2D convolutional neural networks are an essential tool for object detection and classification in the image analysis community, but it seems like the time-frequency community is lagging behind slightly. We believe that these types of networks could be the future of classification using time-frequency distributions; it seems too good to not explore further.

5.4 Covariance with a simple classifier

We created covariance matrices, see Eq. (2.33), for every trial on every subject to study how different parts of the brain had similar activities. We can see this as an extremely primitive version of CSP, which should theoretically also exploit patterns in

Table 5.3 The result of running CNN2D networks on high resolution time-frequency transforms. See table 4.3 for the network design. The value shown for every transform is from the network that resulted in the highest encoding accuracy.

Classifier	Data	Encoding acc.	Retrieval acc.
CNN2D—v2	Spectrogram	0.62	0.32
CNN2D	Wavelet	0.69	0.35
CNN2D—v2	Wigner	0.63	0.33
CNN2D—v2	Ambiguity	0.64	0.32
CNN2D	Slepian	0.60	0.33

the covariance between channels. We did not gain as much insight as we hoped, but Table 5.4 shows that an SVM has 51 % classification accuracy, so there is information to distinguish the classes in the covariance. In Figure 5.5 we show the average correlation on subject 5 for all channels. It seems that there is high correlation for channels close to each other, which is expected.

Table 5.4 Accuracy of SVM classifier on covariance data.

Classifier	Data	Encoding acc.	Retrieval acc.
SVM	Channel covariance	0.51	0.33

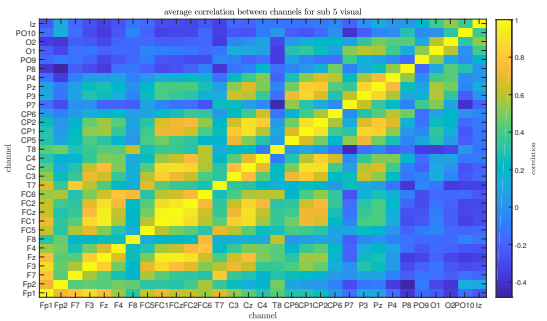

Figure 5.5 Average correlation over all trials for subject 5 on the visual test.

5.5 Common Spatial Patterns with simple classifiers

CSP-based classification is a promising strategy, but unfortunately it did not really rival the other methods in this test, as Table 5.5 shows. It should be mentioned

that CSP has a large amount of exchangeable algorithms and estimators. Essentially every step of CSP can be exchanged for an alternative method while keeping everything else the same and testing the performance. As an example, the simple class covariance matrix estimation technique was proposed at the latest in 1990, and since then an almost innumerable amount of alternative algorithms have been proposed to do it in a better way. The method used in this work is more or less the default setting for CSP in the MNE-Python toolbox. The results were not really overwhelming but definitely significantly above guessing the class, and with for example more modern covariance estimation and a fine-tuned neural network the CSP-based algorithm could definitely be comparable to the other methods. Another, arguably "softer", reason for investigating this method further is that the features stemming from this algorithm, in the form of spatial patterns, are nicely set up for interpretation as they show which patterns separate classes the most.

It really seems like the CSP method is invariant to changes both in covariance estimation method and classifying algorithm. If anything, the encoding accuracy is slightly higher for a larger amount of components as seen in Table 5.5, but the improvement is minimal. The retrieval accuracy stays at a steady 33–34 % for all classifiers, just like all tests so far. It is surprising and slightly disappointing that simple channel covariance fed into an SVM as seen in Section 5.4 produces as good results as this method. However, this version of CSP is also the most primitive, and arguably, more sophisticated CSP algorithms should probably be tested before discarding this method completely.

Figure 5.6 shows the results from training classifiers on CSP features from smaller time bins in encoding data and sequentially using those to classify on time bins in the retrieval data. Much like Figure 5.4 there seems to be no significant correlation between the different times the algorithms train or classify. Everything seems to be nicely distributed around 33.3 %.

Figures 5.7 and 5.8 show the first spatial pattern (that is, the one with the highest classifying power) for Subject 5 and averaged over subjects, respectively. Our expectation was that there would be similarities and a steady evolution over time for the different components, but this definitely does not seem to be the case. It is possible that CSP does not provide meaningfully interpretable features at this level of classification, even though it has been used to a great extent in BCI literature.

5.6 Encoding and retrieval data swapped

One conclusion that is clear and unmistakable is that none of the networks manage to classify the retrieval data at all after training on the encoding data. As explained in chapter 1, some of the generators activating during encoding should also activate during retrieval, but there are probably more since there is more stimulation in the visual cortex. A novel idea for finding mutual features and training a good classifier is to try the reverse problem—training on retrieval and classifying on encoding. Un-

Table 5.5 Encoding and retrieval accuracies on CSP transformed data with simple classifiers. The different parameter options are explained in Section 4.2.

Classifier	Data	Encoding acc.	Retrieval acc.
LDA	CSP	0.48	0.33
SVM	CSP	0.48	0.33
MLP	CSP	0.47	0.33
LDA	CSP with norm. trace	0.49	0.34
SVM	CSP with norm. trace	0.47	0.33
MLP	CSP with norm. trace	0.48	0.34
LDA	CSP with epoch cov. est.	0.47	0.34
SVM	CSP with epoch cov. est.	0.48	0.34
MLP	CSP with epoch cov. est.	0.47	0.33
LDA	CSP, 8 components	0.51	0.34
SVM	CSP, 8 components	0.49	0.34
MLP	CSP, 8 components	0.51	0.33

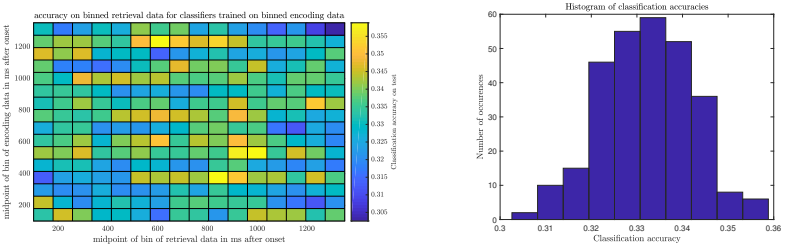

Figure 5.6 (Left) Classification accuracy for CNN1D networks trained on binned encoding data tested on binned retrieval data. Each classifier was trained on *bins* of 240 consecutive time instances (469 ms) from the encoding data with the distances between the midpoints of each *bin* being 60 time instances (117 ms). Testing was done in the same fashion with *bins* of size 240 time instances and a distance of 60 time instances in between, but now using the retrieval data. Both the retrieval and encoding data went through CSP before going into the classifier. (Right) Histogram over all the values in the matrix. The mean classification accuracy is 33.22 %.

fortunately, for this exact data set, this turned out to be unfruitful. Table 5.6 shows the results of a "reverse" test, and the test accuracies on the retrieval data are not significantly higher than guessing the class. More interesting, however, is that the validation accuracies on the retrieval data are not significantly better than guessing, either. This seems to indicate that the retrieval data is extremely hard, if not impossible, to decode and classify in the first place, even with very general methods which work very successfully on encoding data. Despite these results, the method of "reversing" the encoding-retrieval classification paradigm would be interesting

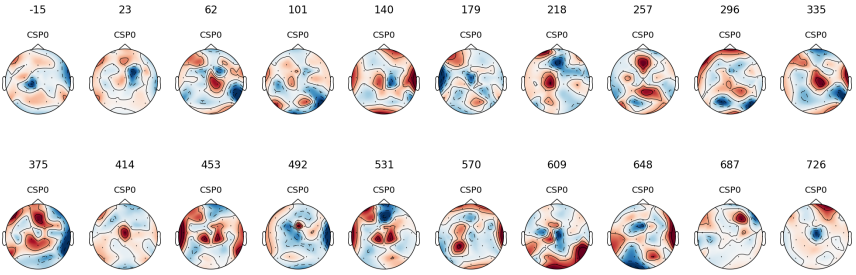

Figure 5.7 The first CSP component in different time bins for subject 5. The number above CSP0 is the time in ms relative to onset of image. The data for every image is 100 time instances (192 ms) and the images have 20 time instances (39 ms) between them.

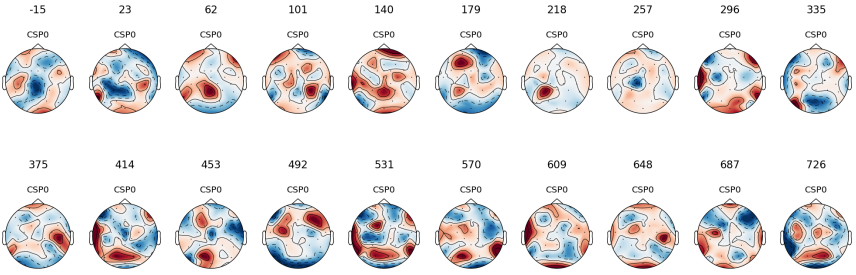

Figure 5.8 The first common spacial pattern over time, average of all subjects. The time is in ms relative to onset of image. The data for every image is 100 time instances (192 ms) and the images have 20 time instances (39 ms) between them.

to investigate further.

Table 5.6 Classifiers trained on the retrieval data from the visual task with 10-fold cross-validation (first number) and then predicting on the encoding data from the visual task (second number). This was to see if there is any structure to the retrieval data at all, and if this structure would be more basic or general and thus have more common parts with encoding data than when training on encoding and testing on retrieval. However, since all of these numbers seem to be no different from randomly guessing, there seems to be barely any structure at all in the retrieval data.

Classifier	Data	10-fold cross validation on retrieval data	Accuracy when testing on encoding data
SVM	Raw cut	0.34	0.33
CNN1D	Raw cut	0.32	0.34
EEGNet	Raw cut	0.34	0.34
CNN2D	Wavelet	0.31	0.32

6

Conclusions

We have investigated a multitude of feature extraction methods and classification networks. Some of them classify based on the raw EEG time data over the scalp (sometimes decimated or cut), some of them utilise time-frequency distributions, and some only use variance changes across channels. Simply looking at validation accuracies we see that higher values come from methods that utilise the time dependency in some way, with 1D convolutional networks reaching 82 % accuracy on the raw data. Qualitatively, however, it seems like decimation with factors up to 16 does not change the validation accuracy much, indicating that the highest frequencies do not really add much information about which type of image the subject was shown.

Time-frequency distributions as a feature extraction method worked decently, reaching an average validation accuracy of almost 70 % over all subjects, but not as well as the raw data. The common spatial patterns algorithm had an even lower validation accuracy, staying at around 50 % for most choices of parameters.

A common theme for all classifying schemes was that the test accuracy was not significantly higher than guessing for any combination of features and networks. The low accuracy could be interpreted as the classifiers being too strict or overtrained. However, since we tried training and classifying on retrieval data with no luck, we deem the retrieval data to be the source of low test accuracy rather than overfitting.

The main difficulty in this work was that it is not immediately obvious which class the trial belongs to from just looking at raw data. In contrast, in e.g. speech recognition in sound recordings or object detection in images, an expert (or in some cases even laymen) can directly say what the correct answer is with a proper display method (i.e. playback of sound recordings or visualisation of images), and the aim is rather to teach the computer to experience what we are experiencing. We are essentially fumbling in the dark, looking for features, and for this reason the method with the best validation accuracy is not necessarily the one to keep testing. CSP seems promising due to the ease of visualising the components of maximum class separation directly. Since all experiments show that time-dependent features are important, the relatively new field of time-dependent common spatial patterns (TDCSP) could be a more valuable step forward than pruning the raw data to get

a higher validation accuracy from the admittedly reliable 1D convolutional neural networks. The theory that fewer neural generators should activate in the brain during retrieval than encoding legitimises the "reversing" scheme where classifiers are trained on retrieval data and tested on encoding as a novel way of extracting common features between the two paradigms.

7

Future ideas

This can probably be said for most investigations of this type, but there are a great many tweaks to the algorithms that we would like to test but did not find the time to. For example, further refining the CNN2D parameters to get better accuracies for this type of EEG data seems like an especially good idea since it has proven to be a reliable method for image analysis in similar fields. Also, variations on the basic CSP algorithm, such as filter bank common spatial patterns (FBCSP), time-dependent common spatial patterns (TDCSP), and common spatio-spectral patterns (CSSP) among others, would definitely be interesting to implement since we believe that the biggest drawback of the CSP algorithm as implemented in this work lacks temporal and spectral information. Testing some type of spatial covariance model from the field of spatial statistics to incorporate spatial dependence would also be interesting, and with both dense EEG setups and better noise suppression/artefact detection, methods from spatial statistics could prove useful to this type of data as well as fMRI, where it is already used today.

It could be interesting to investigate different methods of encoding the spatial dependence of the channels, maybe create an interpolation of values to get a spatial matrix (good for convolutional networks) that evolves through time (good for LSTM) and create a convolutional network with LSTM units for training. Exploring more ensemble-techniques to combine many classifiers could probably push the accuracy more, but would not give new insights into the interpretation of the data.

Bibliography

Anderson, R. (2017). *Modelling and Inference using Locally Stationary Processes: Biomedical applications*. Licentiate Thesis. Lund University, Faculty of Science, Centre for Mathematical Sciences, Mathematical Statistics. URL: http://portal.research.lu.se/portal/files/35026341/thesis_Anderson.pdf.

Barachant, A., S. Bonnet, M. Congedo, and C. Jutten (2010). "Riemannian geometry applied to BCI classification". In: Vigneron, V. et al. (Eds.). *Latent Variable Analysis and Signal Separation*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 629–636. ISBN: 978-3-642-15995-4.

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". *IEEE transactions on neural networks* **5**:2, pp. 157–166.

Blankertz, B., R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Müller (2008). "Optimizing spatial filters for robust EEG single-trial analysis". **25**, pp. 41–56.

Boashash, B. (2013). *Time-Frequency Signal Analysis and Processing: A Comprehensive Review*. Academic Press. ISBN: 9780123984999.

Boashash, B., G. Azemi, and N. A. Khan (2015). "Principles of time-frequency feature extraction for change detection in non-stationary signals: applications to newborn EEG abnormality detection". *Pattern Recognition* **48**:3, pp. 616–627. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.08.016. URL: http://www.sciencedirect.com/science/article/pii/S0031320314003306.

Boashash, B. and S. Ouelha (2018). "Designing high-resolution time-frequency and time-scale distributions for the analysis and classification of non-stationary signals: a tutorial review with a comparison of features performance". *Digital Signal Processing* **77**. Digital Signal Processing & SoftwareX - Joint Special Issue on Reproducible Research in Signal Processing, pp. 120–152. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2017.07.015. URL: http://www.sciencedirect.com/science/article/pii/S1051200417301653.

Bouckaert, R. R. and E. Frank (2004). "Evaluating the replicability of significance tests for comparing learning algorithms". In: Dai, H. et al. (Eds.). *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 3–12. ISBN: 978-3-540-24775-3.

Bramão, I. and M. Johansson (Submitted for publication, 2018). "Neural pattern classification tracks transfer-appropriate processing in episodic memory".

Brynolfsson, J. and M. Sandsten (2017). "Classification of one-dimensional non-stationary signals using the Wigner-Ville distribution in convolutional neural networks". In: *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 326–330. DOI: 10.23919/EUSIPCO.2017.8081222.

C. Van Essen, D., C. H. Anderson, and D. Felleman (1992). "Information processing in the primate visual system: an integrated systems perspective". **255**, pp. 419–23.

Chan, S.-C. and K.-L. Ho (1990). "Efficient computation of the discrete Wigner-Ville distribution". In: *IEEE International Symposium on Circuits and Systems*, 2165–2168 vol.3. DOI: 10.1109/ISCAS.1990.112263.

Choi, H. I. and W. J. Williams (1989). "Improved time-frequency representation of multicomponent signals using exponential kernels". *IEEE Transactions on Acoustics, Speech, and Signal Processing* **37**:6, pp. 862–871. ISSN: 0096-3518. DOI: 10.1109/ASSP.1989.28057.

Chollet, F. et al. (2015). *Keras*. https://keras.io.

Clevert, D.-A., T. Unterthiner, and S. Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)". *arXiv preprint arXiv:1511.07289*.

Cohen, L. (1989). "Time-frequency distributions–a review". *Proceedings of the IEEE* **77**:7, pp. 941–981. ISSN: 0018-9219. DOI: 10.1109/5.30749.

Continuous Wavelet Transform. (N.d.). Online. Accessed: 2018-05-21. URL: https://archive.cnx.org/contents/bcd60908-bf21-452c-bbd1-57bbb8074594@15/continuous-wavelet-transform#uncertainty.

Dahl, G. E., T. N. Sainath, and G. E. Hinton (2013). "Improving deep neural networks for LVCSR using rectified linear units and dropout". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 8609–8613.

Dalin-Volsing, S. (2015). *Classification of Semantic Memories Using Multitaper Spectral Estimation*. Bachelor's Thesis. Centre for Mathematical Sciences, Mathematical Statistics, Lund University, Sweden.

Daubechies, I. (2006). "The wavelet transform, time-frequency localization and signal analysis". *IEEE Trans. Inf. Theor.* **36**:5, pp. 961–1005. ISSN: 0018-9448. DOI: 10.1109/18.57199. URL: https://doi.org/10.1109/18.57199.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. ISBN: 0-89871-274-2.

Daubechies, I. (1993). "Orthonormal bases of compactly supported wavelets II: variations on a theme". *SIAM J. Math. Anal.* **24**:2, pp. 499–519. ISSN: 0036-1410. DOI: 10.1137/0524031. URL: http://dx.doi.org/10.1137/0524031.

Dekker, A. J. den and J. Sijbers (2005). "Advanced image processing in magnetic resonance imaging". In: Landini, L. (Ed.). *Series: Signal Processing and Communications*. Vol. 27. ISBN: 0824725425. Marcel Dekker. Chap. 4, pp. 85–143.

Freeman, W. J. (1975). *Mass action In the nervous system*. Academic Press. URL: http://sulcus.berkeley.edu/MANSWWW/MANSWWW.html.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition (2nd Ed.)* Academic Press Professional, Inc., San Diego, CA, USA. ISBN: 0-12-269851-7.

Gers, F. A. and E. Schmidhuber (2001). "LSTM recurrent networks learn simple context-free and context-sensitive languages". *IEEE Transactions on Neural Networks* **12**:6, pp. 1333–1340.

Gómez, C., F. Vaquerizo-Villar, J. Poza, S. J. Ruiz-Gómez, M. A. Tola-Arribas, M. Cano, and R. Hornero (2017). "Bispectral analysis of spontaneous EEG activity from patients with moderate dementia due to Alzheimer's disease". In: *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Jeju Island, South Korea, July 11-15, 2017*, pp. 422–425. DOI: 10.1109/EMBC.2017.8036852. URL: https://doi.org/10.1109/EMBC.2017.8036852.

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

Grosse-Wentrup, M. and M. Buss (2008). "Multi-class common spatial pattern and information theoretic feature extraction". *IEEE Transactions on Biomedical Engineering* **55**:8, pp. 1991–2000.

Grosse-Wentrup, M. (2008). *Feature Extraction in non-invasive Brain-Computer Interfaces*. Dissertation. Technische Universität München, München.

Grossmann, A. and J. Morlet (1984). "Decomposition of Hardy functions into square integrable wavelets of constant shape". **15**, pp. 723–736.

Gupta, S. S. (2014). "fMRI for mapping language networks in neurosurgical cases". *The Indian Journal of Radiology & Imaging* **24**:1, pp. 37–43.

H. Jansen, B., J. R. Bourne, and J. W. Ward (1981). "Autoregressive estimation of short segment spectra for computerized EEG analysis". **28**, pp. 630–638.

Hazarika, N., J. Z. Chen, A. C. Tsoi, and A. Sergejew (1997). "Classification of EEG signals using the wavelet transform". *Signal Process.* **59**:1, pp. 61–72. ISSN: 0165-1684. DOI: 10.1016/S0165-1684(97)00038-8. URL: http://dx.doi.org/10.1016/S0165-1684(97)00038-8.

He, K., X. Zhang, S. Ren, and J. Sun (2015). "Delving deep into rectifiers: surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

Heyden, M. (2016). *Classification of EEG data using machine learning techniques*. Master's Thesis TFRT-6019. Department of Automatic Control, Lund University, Sweden.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov (2012). "Improving neural networks by preventing co-adaptation of feature detectors". *arXiv preprint arXiv:1207.0580*.

Hochreiter, S. and J. Schmidhuber (1997). "Long short-term memory". *Neural computation* **9**:8, pp. 1735–1780.

Hornik, K. (1991). "Approximation capabilities of multilayer feedforward networks". *Neural networks* **4**:2, pp. 251–257.

Ihalainen, T., L. Kuusela, S. Turunen, S. Heikkinen, S. Savolainen, and O. Sipilä (2015). "Data quality in fMRI and simultaneous EEG–fMRI". *Magnetic Resonance Materials in Physics, Biology and Medicine* **28**:1, pp. 23–31. ISSN: 1352-8661. DOI: 10.1007/s10334-014-0443-6. URL: https://doi.org/10.1007/s10334-014-0443-6.

Ioffe, S. and C. Szegedy (2015). "Batch normalization: accelerating deep network training by reducing internal covariate shift". *CoRR* **abs/1502.03167**. arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

J. Koles, Z., M. S. Lazar, and S. Z. Zhou (1990). "Spatial patterns underlying population differences in the background EEG". **2**, pp. 275–84.

Jeong, J. and W. Williams (1992). "Kernel design for reduced interference distributions". **40**, pp. 402–412.

Jing, S. and X. Yun (2017). "Off-line analysis of motor imagery electroencephalogram". In: *2017 First International Conference on Electronics Instrumentation Information Systems (EIIS)*, pp. 1–6. DOI: 10.1109/EIIS.2017.8298770.

Kaplan, A. Y., A. A. Fingelkurts, A. A. Fingelkurts, S. V. Borisov, and B. S. Darkhovsky (2005). "Nonstationary nature of the brain activity as revealed by EEG/MEG: methodological, practical and conceptual challenges". *Signal Processing* **85**:11. Neuronal Coordination in the Brain: A Signal Processing Perspective, pp. 2190–2212. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2005.07.010. URL: http://www.sciencedirect.com/science/article/pii/S0165168405002094.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Lacoste, A., F. Laviolette, and M. Marchand (2012). "Bayesian comparison of machine learning algorithms on single and multiple datasets". In: Lawrence, N. D. et al. (Eds.). *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*. Vol. 22. Proceedings of Machine Learning Research. PMLR, La Palma, Canary Islands, pp. 665–675. URL: http://proceedings.mlr.press/v22/lacoste12.html.

Larson, E., A. Gramfort, D. A. Engemann, jaeilepp, T. L. Brooks, M. Jas, C. Brodbeck, M. Luessi, jona-sassenhagen, R. Goj, J.-R. KING, wronk, yousrabk, M. van Vliet, C. Holdgraf, A. Leggitt, A. R. Dykstra, R. Trachel, lorenzo-desantis, mbillingr, dgwakeman, D. Strohmeier, T. Linzen, H. Bharadwaj, E. Ruzich, alexandre barachant, cmoutard, C. Bailey, jmontoyam, and C. Brunner (2016). *mne-python: v0.12*. DOI: 10.5281/zenodo.51277. URL: https://doi.org/10.5281/zenodo.51277.

Laton, J., J. V. Schependom, J. Gielen, J. Decoster, T. Moons, J. D. Keyser, M. D. Hert, and G. Nagels (2014). "In search of biomarkers for schizophrenia using electroencephalography". In: *2014 International Workshop on Pattern Recognition in Neuroimaging*, pp. 1–4. DOI: 10.1109/PRNI.2014.6858527.

Lawhern, V. J., A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance (2016). "EEGNet: a compact convolutional network for EEG-based brain-computer interfaces". *arXiv preprint arXiv:1611.08024*.

LeCun, Y., L. Bottou, G. B. Orr, and K.-R. Müller (1998). "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, pp. 9–50.

Lindgren, G., H. Rootzén, and M. Sandsten (2014). *Stationary stochastic processes for scientists and engineers.* Boca Raton : CRC Press, cop. 2014. ISBN: 9781466586185.

Lystad, R. P. and H. Pollard (2009). "Functional neuroimaging: a brief overview and feasibility for use in chiropractic research". *The Journal of the Canadian Chiropractic Association* **53**:1, pp. 59–72. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2652631/.

Maas, A. L., A. Y. Hannun, and A. Y. Ng (2013). "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1, p. 3.

Mallat, S. G. (1989). "A theory for multiresolution signal decomposition: the wavelet representation". *IEEE Trans. Pattern Anal. Mach. Intell.* **11**:7, pp. 674–693. ISSN: 0162-8828. DOI: 10.1109/34.192463. URL: http://dx.doi.org/10.1109/34.192463.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul

Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: large-scale machine learning on heterogeneous systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Martínez, A. M. and A. C. Kak (2001). "PCA versus LDA". *IEEE transactions on pattern analysis and machine intelligence* **23**:2, pp. 228–233.

Müller-Gerking, J., G. Pfurtscheller, and H. Flyvbjerg (1999). "Designing optimal spatial filters for single-trial EEG classification in a movement task." *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology* **110 5**, pp. 787–98.

Muthuswamy, J. and N. V. Thakor (1998). "Spectral analysis methods for neurological signals". *Journal of Neuroscience Methods* **83**:1, pp. 1–14. ISSN: 0165-0270. DOI: https://doi.org/10.1016/S0165-0270(98)00065-X. URL: http://www.sciencedirect.com/science/article/pii/S016502709800065X.

Nadeau, C. and Y. Bengio (2003). "Inference for the generalization error". *Machine Learning* **52**:3, pp. 239–281. ISSN: 1573-0565. DOI: 10.1023/A:1024068626366. URL: https://doi.org/10.1023/A:1024068626366.

Ohlsson, M. (2017). *Lecture notes on introduction to artificial neural networks and deep learning (fytn14/extq40*.

Oostenveld, R., P. Fries, E. Maris, and J.-M. Schoffelen (2011). "FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data". *Intell. Neuroscience* **2011**, 1:1–1:9. ISSN: 1687-5265. DOI: 10.1155/2011/156869. URL: http://dx.doi.org/10.1155/2011/156869.

Oppenheim, A. V. and R. W. Schafer (2009). *Discrete-Time Signal Processing*. 3rd. Prentice Hall Press, Upper Saddle River, NJ, USA. ISBN: 9780131988422.

Pardey, J., S. Roberts, and L. Tarassenko (1996). "A review of parametric modelling techniques for EEG analysis". *Medical Engineering & Physics* **18**:1, pp. 2–11. ISSN: 1350-4533. DOI: https://doi.org/10.1016/1350-4533(95)00024-0. URL: http://www.sciencedirect.com/science/article/pii/1350453395000240.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: machine learning in Python". *Journal of Machine Learning Research* **12**, pp. 2825–2830.

Polyn, S. M., V. S. Natu, J. D. Cohen, and K. A. Norman (2005). "Category-specific cortical activity precedes retrieval during memory search". *Science* **310**:5756, pp. 1963–1966. ISSN: 0036-8075. DOI: 10.1126/science.1117645. eprint:

http://science.sciencemag.org/content/310/5756/1963.full.pdf. URL: http://science.sciencemag.org/content/310/5756/1963.

Refaeilzadeh, P., L. Tang, and H. Liu (2009). "Cross-validation". **532–538**, pp. 532–538.

Roach, B. J. and D. H. Mathalon (2008). "Event-related EEG time-frequency analysis: an overview of measures and an analysis of early gamma band phase locking in schizophrenia". *Schizophrenia Bulletin* **34**:5, pp. 907–926. DOI: 10.1093/schbul/sbn093. eprint: /oup/backfile/content_public/journal/schizophreniabulletin/34/5/10.1093/schbul/sbn093/2/sbn093.pdf. URL: http://dx.doi.org/10.1093/schbul/sbn093.

Rogers, B. P., J. D. Carew, and M. Meyerand (2004). "Hemispheric asymmetry in supplementary motor area connectivity during unilateral finger movements". *NeuroImage* **22**:2, pp. 855–859. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2004.02.027. URL: http://www.sciencedirect.com/science/article/pii/S105381190400120X.

Russell, S. J. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education. ISBN: 0137903952.

Rutkowski, G., K. Patan, and P. Leśniak (2013). "Comparison of time-frequency feature extraction methods for EEG signals classification". In: Rutkowski, L. et al. (Eds.). *Artificial Intelligence and Soft Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 320–329. ISBN: 978-3-642-38610-7.

Ryynanen, O., J. Hyttinen, and J. Malmivuo (2004). "Study on the spatial resolution of EEG - effect of electrode density and measurement noise". In: *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 2, pp. 4409–4412. DOI: 10.1109/IEMBS.2004.1404226.

Sandsten, M. (2018). *Time-Frequency Analysis of Time-Varying Signals and Non-Stationary Processes. An Introduction*. Mathematical Sciences, Lund University.

Slepian, D. (1978). "Prolate spheroidal wave functions, Fourier analysis, and uncertainty—V: the discrete case". *The Bell System Technical Journal* **57**:5, pp. 1371–1430. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1978.tb02104.x.

Waldhauser, G. T., V. Braun, and S. Hanslmayr (2016). "Episodic memory retrieval functionally relies on very rapid reactivation of sensory information". *Journal of Neuroscience* **36**:1, pp. 251–260. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.2101-15.2016. eprint: http://www.jneurosci.org/content/36/1/251.full.pdf. URL: http://www.jneurosci.org/content/36/1/251.

Xu, Y., S. Haykin, and R. J. Racine (1999). "Multiple window time-frequency distribution and coherence of EEG using Slepian sequences and Hermite functions". *IEEE Transactions on Biomedical Engineering* **46**:7, pp. 861–866. ISSN: 0018-9294. DOI: 10.1109/10.771197.

Yger, F., F. Lotte, and M. Sugiyama (2015). "Averaging covariance matrices for EEG signal classification based on the CSP: an empirical study". In: *EUSIPCO 2015*. Nice, France. URL: https://hal.inria.fr/hal-01182728.