Imperial College London

Department of Computing

Deep learning for automated sleep monitoring

Orestis Tsinalis

Thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computing of Imperial College London, June 2016

Declaration of originality

I hereby declare that this thesis entitled 'Deep learning for automated sleep monitoring' is my own work. The work of others has appropriately been acknowledged.

Orestis Tsinalis, June 2016

Copyright declaration

Abstract

Wearable electroencephalography (EEG) is a technology that is revolutionising the longitudinal monitoring of neurological and mental disorders, improving the quality of life of patients and accelerating the relevant research. As sleep disorders and other conditions related to sleep quality affect a large part of the population, monitoring sleep at home, over extended periods of time could have significant impact on the quality of life of people who suffer from these conditions. Annotating the sleep architecture of patients, known as sleep stage scoring, is an expensive and time-consuming process that cannot scale to a large number of people. Using wearable EEG and automating sleep stage scoring is a potential solution to this problem. In this thesis, we propose and evaluate two deep learning algorithms for automated sleep stage scoring using a single channel of EEG. In our first method, we use time-frequency analysis for extracting features that closely follow the guidelines that human experts follow, combined with an ensemble of stacked sparse autoencoders as our classification algorithm. In our second method, we propose a convolutional neural network (CNN) architecture for automatically learning filters that are specific to the problem of sleep stage scoring. We achieved state-of-the-art results (mean F_1-score 84%; range 82–86%) with our first method and comparably good results with the second (mean F_1-score 81%; range 79–83%). Both our methods effectively account for the skewed performance that is usually found in the literature due to sleep stage duration imbalance. We propose a filter analysis and visualisation methodology for CNNs to understand the filters that CNNs learn. Our results indicate that our CNN was able to robustly learn filters that closely follow the sleep scoring guidelines.

Acknowledgements

I would like to thank my supervisor, Dr. Stefanos Zafeiriou, for believing in my work and supporting me immensely on many levels. It was a great pleasure working with him. I would also like to thank Professor Paul Matthews, without whom this thesis would have not been possible. I am grateful that I had the chance to work with him.

I would like to thank my colleagues Akara, Axel, Bertan, Chao, David, Diana, Dilshan, Evann, Florian and Ibrahim for the good and creative times we had together. I would also like to thank the providers of the Physionet dataset on which this work is based.

My friends Max, Sne, Mihalis, Yannis, Maria, Phaedon and Ioanna supported me when I most needed it, and for that I will forever be grateful. My sister, Myrto, and my family were always there for me.

I dedicate this thesis to Sofia.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Clinical-grade longitudinal monitoring of human sleep could have a significant positive impact on the lives of people suffering from sleep disorders and diseases associated with sleep quality. It could also significantly accelerate the relevant research in sleep science by giving researchers access to larger patient populations and more data per patient.

To monitor their sleep, currently patients need to go to a specialised sleep clinic, and wear a multitude of sensors on their body. Human experts are a core part of a session at a sleep clinic, both for ensuring that all sensors are properly placed and for manually annotating the sensor data after the session is over. Sleep monitoring is therefore an expensive and time-consuming process. Because of this, a patient can usually be monitored for only two nights at a time. Unfortunately, depending on the condition that the patient suffers from, the information derived from these two nights may not be representative of the patient's sleep quality.

With the emergence of wearable sensors it has become possible to conveniently monitor a wide range of physiological signals. Sleep monitoring relies chiefly on the interpretation of electrical activity generated by the brain by placing sensors on the surface of the scalp. These sensor technologies are collectively called *electroencephalography*, or *EEG*. Current EEG technology makes it possible to acquire and transfer brain signals wirelessly. For an EEG device to be wearable and suitable for sleep monitoring, it must be easy to wear

without expert supervision, be stable, and have sufficiently long battery life. All these requirements could be fulfilled by reducing the number of EEG sensors that are used.

The second requirement for an automated sleep monitoring system would be automated annotation of the data. This annotation, called sleep stage scoring (or sleep staging, or sleep scoring), is now done by human experts inspecting visually EEG and other physiological signals. Sleep stage scoring is a time-consuming process, and a major bottleneck for the automation of sleep monitoring.

In this thesis we combine the two requirements above, and investigate automated sleep stage scoring using a single channel of EEG. To achieve this we employed *deep learning*, a family of machine learning algorithms, which recently had tremendous success in domains such as computer vision and natural language processing.

1.1 Contributions

In our work we achieved state-of-the-art results in automated sleep stage scoring using a single channel of EEG, avoiding skewed performance in favour of the most represented sleep stages, which is common in the literature. We proposed two methods. The first is based on time-frequency analysis with hand-engineered features that are fine-tuned to capture sleep stage-specific signal characteristics that the human sleep scorers identify in EEG signals. The second method is a convolutional neural network (CNN) architecture for automatically learning the filters without utilising prior knowledge of the sleep scoring problem. By analysing and visualising the filters that the neural network learns, we show that the filters are related to the sleep scoring guidelines that human experts follow. This is an important finding, because it shows that CNNs, which are commonly viewed as 'black boxes', are in fact interpretable.

1.2 Publications

Our work is summarised in two papers, the first published in the journal *Annals of Biomedical Engineering*, and the second in preparation for the journal *IEEE Transactions on Biomedical Engineering* (preprint available online on arXiv) at the time of writing:

- **Tsinalis, O.**, Matthews, P. M. and Guo, Y. 2016. Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders. *Annals of Biomedical Engineering*, 44(5), pp. 1587–1597. [94]

- **Tsinalis, O.**, Matthews, P. M., Guo, Y. and Zafeiriou, S. 2016. Automatic sleep stage scoring with single-channel EEG using convolutional neural networks. *arXiv preprint arXiv:1610.01683*. [95] (in preparation)

1.3 Thesis structure

The thesis is structured as follows:

Chapter 2: In this chapter we give the background on sleep, sleep science, sleep disorders, wearable electroencephalography, (automated) sleep stage scoring, and the data that we used in our experiments.

Chapter 3: In this chapter we give the background on signal processing. The focus of this chapter is on convolution and time-frequency analysis.

Chapter 4: In this chapter we give a detailed description of deep neural networks. We begin with an overview of the general machine learning background on supervised and unsupervised learning, cross-validation, and overfitting. We then describe how deep neural networks can be built from simpler units, starting with logistic and softmax regression, and continuing with single-layer neural networks and, finally, deep neural networks. We describe prediction (forward propagation), learning (stochastic gradient descent), and partial derivatives computation (backpropagation) in deep neural networks. The chapter concludes with a section on regularisation as a method for avoiding overfitting, and

stacked sparse autoencoders, a deep neural network variant that involves unsupervised pre-training.

Chapter 5: In this chapter we give a detailed description of convolutional neural networks (CNNs). We describe the connection between CNNs and hand-engineered features for classification and give the motivation behind the development of CNNs. After an overview of the use of CNNs in biomedical engineering, we proceed with a description of the minimal CNN, pooling layers, and deep CNNs. We conclude the chapter with a description of forward propagation and backpropagation for CNNs with one-dimensional inputs.

Chapter 6: In this chapter we present our work on automated sleep scoring using time-frequency analysis and stacked sparse autoencoders. This is a work in which we combined hand-engineered features with deep learning to achieve state-of-the-art results in sleep scoring using a single channel of EEG.

Chapter 7: In this chapter we present our work on automated sleep scoring using convolutional neural networks. We show that our algorithm can automatically learn features relevant to the problem of sleep scoring without utilising prior knowledge about the problem as in Chapter 6. In this chapter we show how the filters that are learned by a CNN can be analysed and visualised given a specific classification problem such as sleep scoring.

Chapter 8: In this chapter we conclude this thesis with a summary of the contributions of the work, and the potential avenues for future work.

Chapter 2

Sleep and sleep stage scoring

In this chapter we give the required background on sleep and sleep science. We then describe the 'gold standard' process for sleep monitoring, called polysomnography (PSG), with a particular focus on sleep staging. Next, we describe how PSG and other sleep monitoring methods are used for the diagnosis of sleep disorders. We then describe the recent developments in wearable electroencephalography (EEG) which can lead to affordable and portable sleep monitoring. We close this chapter with a review of the literature on automated sleep staging using single-channel EEG.

2.1 Sleep

Sleep is central to human health. The health consequences of reduced sleep, abnormal sleep patterns or desynchronized circadian rhythms can be emotional, cognitive, or somatic [100]. Disruption of normal sleep patterns can negatively affect cognitive abilities, memory, reaction time, productivity, and creativity [64]. Associations between disruption of normal sleep patterns and neurodegenerative diseases are well recognised [100]. Recent research suggests that detection of sleep/circadian disruption could be a valuable marker of vulnerability and risk in the early stages of neurodegenerative diseases, such as Alzheimer's disease, Parkinson's disease and multiple sclerosis, and that sleep stabilisation could improve the patients' quality of life [100]. Bad sleep quality is also associated

with psychiatric disorders, most notably depression [98, Ch. 7], as well as metabolic disorders, such as obesity and diabetes [64, Ch. 7].

Duration is probably the aspect of sleep that most people are familiar with. Lockley and Foster [64, p. 2] note on the duration of human sleep:

> Adults, on average, sleep about 7 hours a night, with 5% sleeping fewer than 5 hours, and 6% sleeping more than 9 hours ... By contrast, some historical reports suggest that we slept significantly longer in the past. During the long nights of winter, sleep probably occurred for extended periods of time with two or sometimes more discrete bouts of sleep separated by intervals of resting wakefulness. In pre-industrial times, we may have slept up to 10 hours a day, depending on the season. Modern day experiments support these ideas: if people are kept on a winter schedule (long nights, short days), they do sleep more than when kept on a summer pattern. If subjects are given very long opportunities to sleep, they will eventually reach a stable sleep duration of about 8.5 hours in young adults, and 7.5 hours in older adults, more than most people currently get. The introduction of electric lighting in the 19th century, and the restructuring of work hours and social schedules caused by industrialization, have meant that our species has become progressively detached from the natural 24-hour cycles of light and dark.
>
> It seems likely that we sleep less now than at any other time in our recent history. Most data collected from industrial nations over the past 50 years show a general decline in sleep in line with the culture of long work hours, more shift-work, long commutes, global communication across multiple time zones, and freedom from many economic and social constraints.

As Lockley and Foster further note '[t]he natural pattern and duration of human sleep is essentially unknown. ... Urban human sleep patterns are no more natural than those of ... laboratory mice' [64, p. 47].

We have so far used the term *sleep* in a more or less colloquial way, without giving a precise definition of what sleep is. However, in order to *measure* sleep we need a definition of sleep as well as a description of the means by which sleep is measured. It is instructive to start from the behavioural criteria used to define sleep across species. Lockley and Foster give the following behavioural criteria [64, p. 41]:

1. A rapidly reversible state of immobility with greatly reduced sensory responsiveness.
2. Increased arousal thresholds ... and a decreased responsiveness to external stimulation.
3. Species-specific posture and place preference.
4. Behavioural rituals before sleep (e.g. circling, yawning, nest-making).
5. Circadian regulation and persistence of a ∼24-hour rhythm under constant conditions.
6. 'A behaviour that is homeostatically regulated such that lost sleep is associated with an increased drive for sleep, with subsequent "sleep rebound".

Understanding the *reasons* for sleep could have significant impact on treating sleep disorders. As Lockley and Foster note '[w]e don't know exactly why we sleep' [64, p. 40]. There are currently three theories regarding the evolutionary reasons for sleep: (i) cellular restoration, (ii) energy conservation, and (iii) consolidation of memory and learning [64, pp. 40-54].

2.2 A brief history of sleep science

The study of sleep has a long history that goes back at least 2,500 years with the Greek philosophers and physicians Alcmaeon, Hippocrates, and Aristotle, who wrote a treatise titled 'On Sleep and Sleeplessness' around 350 BC [64, Ch. 1]. The first scientist to

Table 2.1: Frequency bands (modified from [20, Ch. 3])

Frequency band	**Frequency Range**
Delta	0–4 Hz
Theta	4–8 Hz
Alpha	8–13 Hz
Beta	15–30 Hz
Lower gamma	30–80 Hz
Upper gamma	80–150 Hz

suggest that the brain is central to sleep was René Descartes in the early 17th century, and, in the early 19th century, Luigi Rolando and Jean Pierre Flourens developed this idea with experiments on birds [64, Ch. 1].

Modern sleep science developed as a result of the technological advances that offered the capability of measuring brain activity during sleep. The first studies [16] in which electrical activity from the surface of the brain was recorded were performed in animals in 1875 by Richard Caton [49]. The first electroencephalogram (EEG) in humans was recorded by Hans Berger in 1924 [21], who developed Caton's methods for measurement in humans [64, Ch. 2]. Berger [10] demonstrated differences between sleep and wakefulness using external scalp recordings [49]. 'Berger was able to show that when subjects were awake, the EEG showed a fast (high frequency) and small amplitude pattern of activity which became slower (lower frequency) and had larger amplitude waves as individuals drifted towards sleep' [64, pp. 7-8].

The first study to classify what is now known as non-REM sleep into five stages based on differences in the electrical activity of the brain using scalp EEG recordings was [65] by Loomis, Harvey and Hobart in 1937. Blake and Gerard in 1937 [13] observed that delta waves (high amplitude <4 Hz activity) are associated with how deep (as determined by response to an auditory stimulus) sleep is in a subject at a particular time, and Davis *et al.* in 1938 [25] described the first part of sleep in more detail [12].

Rapid eye movements (REM) were first studied by Aserinksy and Kleitman [4, 5].

REM sleep was first formally introduced as a sleep stage associated with rapid eye movements by Dement and Kleitman in 1957 [27]. The first sleep scoring manual which formed the basis for the American Academy of Sleep Medicine (AASM) sleep scoring manual [49] that experts follow today was developed and published under the direction of Rechtschaffen and Kales in 1968 [81]. As Edinger and Morin note [30, p. 363], 'it was not until the 1960s and 1970s when researchers increasingly began studying individuals with contrasting forms of sleep/wake pathology including those with chronic nighttime sleep difficulties, those with excessive daytime sleepiness, and those with unusual sleep-related behaviours'.

2.3 Polysomnography and sleep staging

In humans sleep is defined by distinct patterns extracted from electrophysiological signals—signals that capture the electrical activity at different parts of the human body. The central electrophysiological signal for the definition of sleep is the electroencephalogram (EEG), which measures electrical activity on the scalp by attached electrodes. In Figure 2.1 we show 21 electrode positions and their names according to the International 10-20 system.

EEG electrodes record voltage changes from the surface of the brain, which are produced by the electrical and chemical (neurotransmitter) signals that the brain's neurons (nerve cells) exchange with each other. As the brain is assembled of approximately 100 billion neurons, the electrical activity that is recorded from the scalp electrodes is the result of collective interactions of populations of neurons [64, Ch. 2]. 'In general, EEG is not well suited for studies in which precise functional localization is important' [20, p. 16].

The unit of measurement of EEG is volts (more specifically, microvolts or μV). The values of EEG signals are relative values. Effectively, each EEG signal from a particular electrode is the difference in voltage between the electrode at that particular location and a reference electrode placed at a different location (or the average voltage across multiple

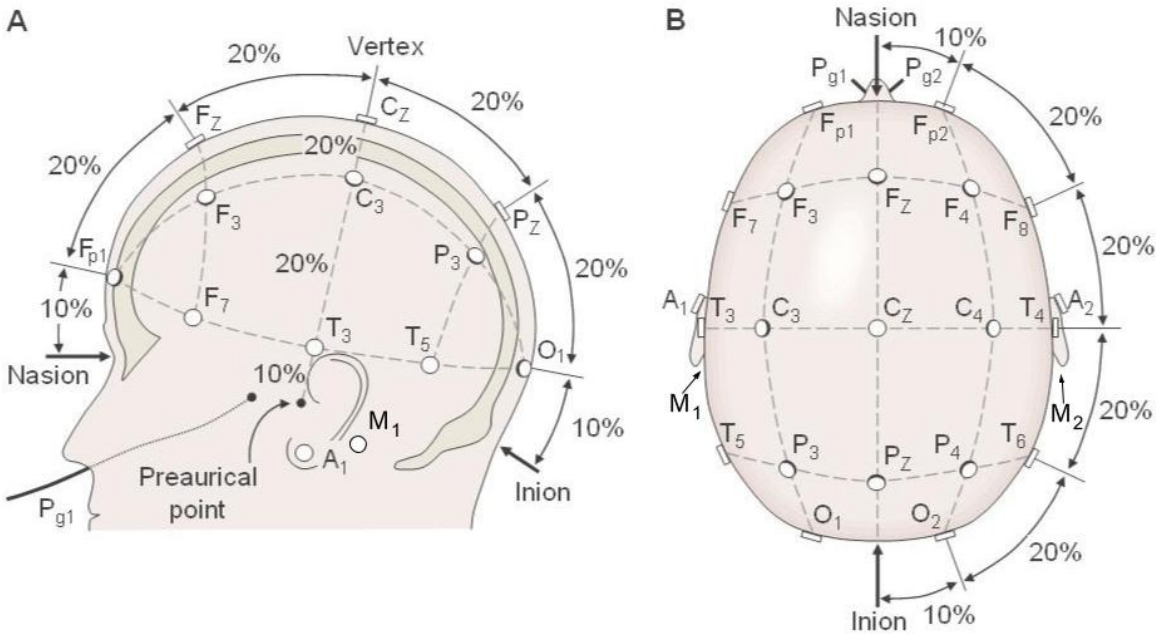

Figure 2.1: Electrode locations according to the International 10-20 system of electroencephalography (EEG) electrode placement. Image downloaded from `http://www.diytdcs.com/tag/1020-positioning/`, and edited to include the mastoid electrodes M_1 and M_2.

electrodes) [20, p. 17]. The EEG derivations (electrode-reference pairs) recommended by AASM for sleep stage scoring are F_4-M_1, C_4-M_1 and O_2-M_1, or, alternatively F_z-C_z, C_z-O_z and C_4-M_1 [49, p. 23].

To measure sleep, EEG is complemented by the electrooculogram (EOG), which measures eye movements through electrodes on the forehead, and the electromyogram (EMG), which measures muscle activity of the submental muscle through electrodes on the chin. These three sensing modalities are the fundamental components of the polysomnogram (PSG), which means the measurement of multiple physiological sleep variables [98, Ch. 2]. The PSG can also include an electrocardiogram (ECG), which measures the electrical activity of the heart, EMG from leg muscles, measurements of respiratory variables, and measurements of body movements [98, Ch. 2]. To record a PSG, subjects are usually admitted to a specialised sleep clinic, but there are portable systems to record PSGs at home as well. Subjects are usually admitted to the sleep clinic for one or two nights [73].

According to the new American Academy of Sleep Medicine (AASM) manual [49],

sleep is categorised into four stages. These are Rapid Eye Movement (stage R) sleep and 3 Non-REM stages, stages N1, N2 and N3. Formerly, stage N3 (also called Slow Wave Sleep, or SWS) was divided into two distinct stages, Non-REM Stage 3 and Non-REM Stage 4 [81]. To these a Wake (W) stage is added. These stages are defined by the characteristics of the electrical activity recorded from the different modalities in the PSG, mainly the EEG, EOG and chin EMG. After the PSG is recorded, it is divided into 30-second intervals, called *epochs*. Then, one or more experts classify each epoch into one of the five stages (N1, N2, N3, R or W) by quantitatively and qualitatively examining the signals of the PSG in the time and frequency domains. Sleep scoring is performed according to the Rechtschaffen and Kales sleep staging criteria [81]. In Table 2.2 we reproduce the Rechtschaffen and Kales sleep staging criteria [81], merging the criteria for Non-REM Stage 3 and Non-REM Stage 4 into a single stage (N3), following the exposition in [87], [64, Ch. 2], [88, Ch. 1], and [98, Ch. 1]. The '% Time Asleep' gives a range of the percentage of time spent at each sleep stage relative to total time asleep (i.e. excluding awakenings in the middle of sleep) in people with normal sleep.

Apart from the electrophysiological features that formally characterise each sleep stage, the sleep stages are also differentiated from behavioural characteristics [88, Ch. 1]. During stage N1, which is called a transitional stage because most people enter sleep through it, people are often unaware that they are asleep and can sometimes detect or respond to external stimuli. During stage N2 people start being unresponsive to external stimuli. Stage N3 is a stage of deep sleep, as people are very difficult to awaken out of it. Finally, stage R, is also called 'paradoxical sleep', because the brain activity during it is similar to brain activity during wakefulness. However, muscle tone as captured by chin EMG is virtually absent during stage R.

The scoring rules that we see in Table 2.2 refer to the features of the epoch that is to be scored. Additionally, the AASM manual [49] includes a number of rules that recommend taking into account neighbouring epochs for the scoring of each current epoch under certain circumstances. We identified 12 rules in total concerning the transition between

Table 2.2: The Rechtschaffen and Kales sleep staging criteria [81], following the exposition in [87], [64, Ch. 2], [88, Ch. 1], and [98, Ch. 1].

Sleep Stage	**% Time Asleep** [88]	**EEG Features** [87, 98, 64]	**Eye Movements** [98]	**Chin EMG Activity** [98]
W	N/A	Low-amplitude, mixed (2-7 Hz), and sometimes alpha (8-13 Hz) activity for $> 50\%$ of epoch. Beta (15-30) activity can also be present.	Many, varied, usually fast	High
R	20-25%	Low-amplitude, irregular mixed (2-7 Hz) activity, and sometimes saw-toothed waves. Beta (15-30) activity can also be present.	Rapid, jerky, and usually lateral movements in clusters	Virtually absent, occasional short bursts
N1	5-10%	50% of the epoch contains low-amplitude, mainly irregular theta (4-7 Hz) and mixed (2-7 Hz) activity, and triangular vertex waves. $< 50\%$ of the epoch contains alpha (8-13 Hz) activity	Slow, rolling, lateral movements	Slightly lowered
N2	40-50%	Sleep spindles, K complexes, some low-amplitude theta (4-7 Hz) activity, and $< 20\%$ of the epoch may contain high amplitude ($> 75\mu V$) low frequency (<2 Hz) activity	None	Low
N3	20-25%	$20-50\%$ (formerly Non-REM Stage 3) or $> 50\%$ (formerly Non-REM Stage 4) of the epoch consists of high amplitude ($> 75\mu V$), low frequency (<2 Hz) activity. Spindles less prominent, and K complexes longer, and less discrete.	None	Low

Table 2.3: The transition rules summarised from the AASM sleep scoring manual [49, Chapter IV, pp. 23–31].

Stage Pair	Transition Pattern	Rule	Differentiating Features
N1-N2	N1-{N1,N2}	5.A.Note.1	Arousal, K-complexes, sleep spindles
	(N2-)N2-{N1,N2}(-N2)	5.B.1	K-complexes, sleep spindles
		5.C.1.b	Arousal, K-complexes, sleep spindles
	N2-{N1-N1,N2-N2}-N2	5.C.1.c	Alpha, body movement, slow eye movement
N1-R	R-R-{N1,R}-N2	7.B	Chin EMG tone
		7.C.1.b	Chin EMG tone
		7.C.1.c	Chin EMG tone, arousal, slow eye movement
	R-{N1-N1-N1,R-R-R}	7.C.1.d	Alpha, body movement, slow eye movement
N2-R	R-R-{N2,R}-N2	7.C.1.e	Sleep spindles
	(N2-)N2-{N2,R}-R(-R)	7.D.1	Chin EMG tone
		7.D.2	Chin EMG tone, K-complexes, sleep spindles
		7.D.3	K-complexes, sleep spindles

Curly braces indicate choice between the stages or stage progressions in the set based on the distinctive features, and parentheses indicate optional epochs.

certain sleep stage pairs that refer to 7 distinct transition patterns, as shown in Table 2.3. These rules apply to three sleep stage pairs, N1-N2, N1-R and N2-R. The transition patterns include up to two preceding or succeeding neighbouring epochs.

An additional feature that is included in Table 2.3 is an arousal. According to the AASM sleep scoring manual an arousal is scored 'during sleep stages N1, N2, N3, or R if there is an abrupt shift of EEG frequency including alpha, theta and/or frequencies greater than 16 Hz (but not spindles) that last at least 3 seconds, with at least 10 seconds of stable sleep preceding the change. Scoring of arousal during REM requires a concurrent increase in submental EMG lasting at least 1 second' [49, p. 37].

It should finally be mentioned that an additional scoring rule exists for epochs that include a major body movement, defined as a 'movement and muscle artifact obscuring the EEG for more than half an epoch to the extent that the sleep stage cannot be

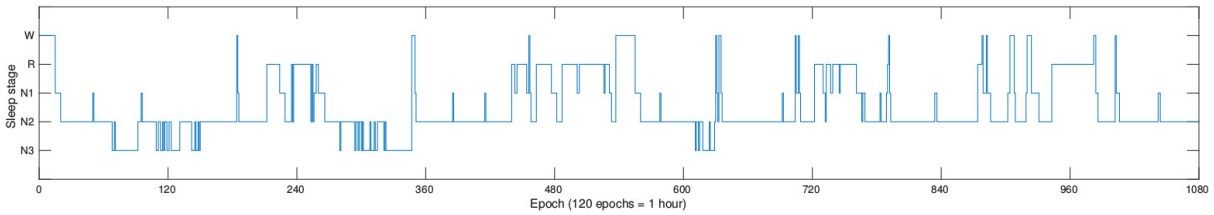

Figure 2.2: An example of a hypnogram of a subject from the Physionet dataset (see Section 2.8.

determined' [49, p. 31]. An epoch that includes a major body movement is scored either as stage W (if alpha activity is present, or an epoch scorable as stage W precedes or follows the epoch), or, otherwise, as the same stage as the epoch that follows it [49, p. 31].

After each epoch of a subject's PSG has been scored as belonging into one of the five sleep stages, a plot of the progression of an entire sleep episode (also called *sleep architecture*) can be created. This plot is called the subject's *hypnogram*. An example of a hypnogram of a subject with normal sleep is shown in Figure 2.2.

As we see in Figure 2.2 there is a certain progression from one sleep stage to the next, and this progression is repeated multiple times during a sleep episode. A single occurrence of this progression, which at a high level is N1-N2-N3-R is called a sleep cycle and has duration of 90-100 minutes. A night of sleep for a subject that has normal sleep corresponds to 4-5 sleep cycles [64, Ch. 2]. In general, as we can see in Figure 2.2, for subjects with normal sleep, stage N3 sleep occurs during the first half of the night, while stage R sleep is more prevalent during the second half of the night [64, Ch. 2]. Subjects can wake up for short periods of time during their sleep.

The hypnogram can be used to calculate a number of variables that are useful for sleep research and for diagnosing sleep disorders. In order to derive these variables from the hypnogram a 'Lights Off' and a 'Lights On' indicator must be included with the hypnogram. The 'Lights Off' indicator marks the beginning of the sleep study. When the subject turns the lights off this is viewed as an indication of the subject's intention to fall asleep [88, Ch. 15]. The 'Lights On' indicator marks the end of the study. In Table

Table 2.4: Variables derived from the hypnogram, some of which are used as sleep quality metrics from [88, Ch. 15] and [98, Ch. 2].

Variable	**Definition / Details**
Time in bed (TIB)	Total time in minutes from 'Lights Off' to 'Lights On'
Total sleep time (TST)	Total time spent asleep, i.e. in one of stages N1, N2, N3 and R
Sleep efficiency (SE)	Percentage of time asleep while in bed: SE = TST / TIB
Total awake time (TWT)	Total time spent awake during study: TWT = TIB - TST
Sleep onset	Time when sleep (stage N2 >1 min) first occurs after 'Lights Off'
Sleep onset latency (SOL)	Time between 'Lights Off' and sleep onset
Wake after sleep onset (WASO)	Time spent awake after sleep onset: WASO = TWT - Sleep latency
Number of awakenings	Lasting for longer than 30 seconds to 2 minutes
Stage R onset	Time at which stage R sleep (>1 min) first occurs after sleep onset
Stage R latency	Time between sleep onset and stage R onset
Time in each sleep stage	Total and sometimes separately for first and second half of night
% time in each sleep stage	As above but as a percentage of total sleep time

2.4 we list some of the most common variables which are included in polysomnography reports, as they are given in [88, Ch. 15] and [98, Ch. 2].

2.4 Sleep disorders and their diagnosis

Sleep disorders are formally classified using three main classification systems (or taxonomies) [30]. These are the *International Classification of Diseases* (ICD) [99] from the World Health Organisation, the *Diagnostic and Statistical Manual of Mental Disorders* (DSM) [3] from the American Psychiatric Association, and the *International Classification of Sleep Disorders* (ICSD) [2] from the American Academy of Sleep Medicine.

As shown in Table 2.5 the ICSD [2] classifies sleep disorders into six major categories: insomnias, sleep-related breathing disorders, central disorders of hypersomnolence, circadian rhythm sleep-wake disorders, parasomnias, and sleep-related movement disorders. These include approximately 60 sub-categories in total (we can see some examples in

Table 2.5: Major categories of sleep disorders summarised from the *International Classification of Sleep Disorders* (ICSD) [2] of the American Academy of Sleep Medicine.

Category	**General Characteristics [98]**	**Examples**
1. Insomnias	Sleep too short, too interrupted, of poor quality, or a combination of these	Psychophysiological insomnia, behavioural insomnia of childhood
2. Sleep-related breathing disorders	Breathing difficulties during sleep	Obstructive sleep apnoea, sleep-related hypoventilation/hypoxemia
3. Central disorders of hypersomnolence (or hypersomnias of central origin)	Excessive daytime sleepiness; central means not due to a circadian rhythm sleep-wake disorder, sleep-related breathing disorder, or other cause of disturbed nocturnal sleep	Narcolepsy due to medical condition, idiopathic hypersomnia with long sleep time
4. Circadian rhythm sleep-wake disorders	Disturbance of the normal sleep-wake rhythm	Jet lag disorder, shift-work disorder
5. Parasomnias	Unusual behaviours that occur during sleep	Sleep walking, sleep terrors, nightmare disorder
6. Sleep-related movement disorders	Abnormal movements during sleep	Restless legs syndrome, periodic limb movement disorder

Table 2.5). 'Sleep problems are common in patients with psychiatric conditions, including depression, anxiety (post-traumatic stress disorder, generalised anxiety disorder, and panic disorder), bipolar disorder, schizophrenia, dementia, and substance abuse' [98, Ch. 7]. In 2.6 we provide a summary of the sleep problems that are associated with psychiatric disorders.

Regarding the validity and reliability of sleep disorder classification systems, Edinger and Morin say: 'Despite limited evidence supporting their validity and reliability, these classification systems remain widely used by clinicians and researchers ... Future research is needed to examine more closely the validity and reliability of various sleep disorders

Table 2.6: Sleep problems associated with psychiatric disorders (adapted from [98, Ch. 7])

Psychiatric disorder	**Associated sleep disorders**
Depression	Insomnia (about 75% of patients), Hypersomnia (5-10 %)
Post-traumatic stress disorder	Insomnia, parasomnias (nightmares, night terrors, sleep-walking)
Generalised anxiety disorder	Onset insomnia (about 20-30%)
Panic disorder	Nocturnal panic attacks, onset insomnia (when fearing these attacks)
Schizophrenia	Onset and maintenance insomnia, free-running circadian rhythm sleep disorder
Dementias	Insomnia, agitation at bedtime, delayed sleep phase circadian rhythm sleep disorder
Withdrawal from opiates or alcohol	Insomnia, irregular sleep-wake circadian rhythm sleep disorder

as well as the clinical utility of current taxonomies' [30, p. 361].

Sleep disorders are affected by multiple psychological, medical, environmental, and circadian factors [30]. Therefore, in order to produce an accurate diagnosis of sleep disorders, sleep clinicians need to perform a comprehensive evaluation of patients. This evaluation should include a clinical interview, screening questionnaires, self-monitoring, polysomnography, multiple sleep latency test (MSLT) and other methods, such as actigraphy [30], [98, Ch. 2] (see Table 2.7) [1].

The methods of evaluation of patients for diagnosing sleep disorders presented in Table 2.7 have different degrees of objectivity, standardisation, and usefulness [30], [98, Ch. 2]. These characteristics affect their potential for being automated, as well as the impact that such automation could have.

According to Edinger and Morin [30, p. 376] '[t]he clinical interview is the most important assessment component for any sleep disorder'. It can be partly standardised

[1]The data that we used in this thesis included exclusively polysomnography.

Table 2.7: The different methods used in a comprehensive evaluation of patients for diagnosing sleep disorders [30], [98, Ch. 2]

Method	**Purpose**
Clinical interview	'[G]ather[ing] detailed information about the presenting sleep complaint, psychiatric and medical history, use of prescribed and over-the-counter medications, history of alcohol and drug abuse, psychosocial and family history, and previous treatments and their outcomes' [30, p. 376]
Screening questionnaires (or psychometric instruments)	General screening, evaluating the severity and impact of sleep disorders, and assessing the presumed mediating factors of a given disorder [30]
Self-monitoring (sleep diary)	'[I]ncludes entries for bedtime, arising time, sleep latency, number and duration of awakenings, total sleep time, naps, use of sleep aids, and various indices of sleep quality and daytime functioning' [30, p. 379]
Polysomnography (PSG)	Overnight recording of electrophysiological signals such as EEG, EOG, and EMG, as well as respiratory and cardiac activity, which is scored by human experts to produce a visualisation of the sleep architecture (hypnogram) of a subject (see Section 2.3)
Multiple sleep latency test (MSLT)	'[D]aytime laboratory test conducted after a full night of PSG recording. During the MSLT, the patient is offered five, 20-minute nap opportunities' [30, p. 377]
Actigraphy	Monitoring rest/activity cycles using a non-invasive method (usually a wrist-worn device which contains an accelerometer that measures movement) [98, Ch. 2]

through various structured interview guides that have been developed based on the different sleep disorder classification systems, 'but they are all designed for use by those who have some knowledge and experience in the area of sleep disorder diagnosis' [30,

Table 2.8: Example of variables monitored by a subject in a sleep diary [98, Ch. 2]

Variable
1. Time you went to bed last night
2. Time you fell asleep (roughly)
3. Time you finally woke up this morning
4. Did the alarm wake you up?
5. Time you got up
6. Number of times you woke up during the night
7. How many hours of sleep you got overall
8. Quality of sleep (out of 10, from 1 = very bad to 10 = very good)
9. Comment (can include morning feeling, medication taken, etc)

p. 376]. Questionnaires are standardised, but subjective. Although there are multiple questionnaires that are used in practice, 'none of [them] have been specifically validated to make a formal diagnosis', as Edinger and Morin note [30, p. 378]. As such, they should not be used in isolation as evidence of a sleep disorder diagnosis. Rather, they should be used as complementary tools to the other clinical and laboratory assessment procedures' [30, p. 378]. Self-monitoring methods, such as sleep diaries, are usually standardised but subjective. Their main advantage is that they offer valuable information for the sleep schedules of patients over extended periods of time [30, 14] (see Table 2.8). Their main disadvantage is that the collected data might be inaccurate [14].

'Polysomnography (PSG) is considered the gold standard for measuring sleep and is essential to confirm the diagnosis of several sleep disorders such as sleep apnoea, periodic limb movements, narcolepsy, and some parasomnias' [30, p. 376] [57]. PSG may also be used to assist in the diagnosis of certain types of insomnia, as in some cases a patient might be having what is called 'sleep misperception', i.e. they might be significantly underestimating the amount of sleep that they are getting [98, Ch. 3]. 'Individuals with insomnia tend to overestimate SOL [sleep onset latency, see Table 2.4] and WASO [wake after sleep onset, see Table 2.4] and to underestimate TST [total sleep time, see Table 2.4]

in comparison to PSG measures', so a PSG gives more reliable data than a sleep diary [14] (see Table 2.4 for the definition of these sleep quality metrics). There has also been research involving the examination of PSG recordings in the context of hypesomnias, such as [35].

PSG studies have also shown that patients with depression exhibit a number of disturbances during their sleep. Increased wakefulness, reduced sleep efficiency, increased sleep onset latency, decreased total sleep time, potentially reduced stage N3 sleep, decreased stage R latency, increased first stage R period duration, and increased stage R density are some of the differences between subjects with depression and normal controls [6, 83]. A PSG study in patients with major depression has shown that 'sleep abnormality was predictive of a lower recovery rate and a higher risk of recurrence' [93]. Patients suffering from neurodegenerative disorders also exhibit abnormal sleep characteristics that can be identified using PSG recordings [100]. For example, patients with Alzheimer's Disease exhibit nocturnal awakenings, altered K complex and sleep spindle patterns, decrease in beta activity in stages R and W, and decrease in stage N3 sleep [100, Supplementary Information].

PSG is standardised and offers largely objective measurements, based on the technical guidelines and sleep scoring rules in the AASM manual [49]. There are, however, cases of low inter-rater agreement among human scorers, to which we will refer in Section 2.6. Also, as Kushida *et al.* [57] note:

> Since the PSG is considered the reference standard, the reliability and technical accuracy of PSG is generally accepted without question. However, PSG, even when accurately measured, recorded, and analysed, may misclassify patients based upon night-to-night variability in measured parameters, the use of different types of leads that may lead to over- or underestimation of events ..., and the vagaries of the clinical definitions of disease.

One of the main problems with PSG recordings recorded in laboratory settings is that the sleep of the subjects is affected by the change in setting [29]. A common characteristic

of laboratory PSG are the so-called 'first night effects', which include 'reduced total sleep times, lower sleep efficiencies, decreased slow-wave sleep, and elevated REM latencies' [29]. As Edinger *et al.* note [29]:

> As a consequence of these FNEs [first night effects] sleep researchers and clinicians have adopted the common practice of ignoring first night data and relying on findings from subsequent nights to address their research and clinical questions ... Even when multiple LPSG [laboratory PSG] recordings can be conducted, sleep data from some types of subjects/patients might be confounded by residual laboratory effects that persist beyond the first night in a short series of nocturnal sleep recordings.

Because of these reasons, 'home [PSG] studies are probably more representative of the participants' typical sleep and sleep-related behaviors, they may require less technician time and cost, and they are less influenced by "first-night effects"' [14].

The multiple sleep latency test (MSLT) is an objective and standardised method of sleep disorder evaluation. 'The MSLT should be an integral component of the evaluation whenever daytime alertness is compromised. It is particularly useful to determine the degree of impairment associated with disorders such as sleep apnoea and narcolepsy' [30, p. 377]. The MSLT should always be preceded by an overnight PSG.

Actigraphy is the process of monitoring movement via wrist-worn devices which contain an accelerometer, i.e. a sensor that measures acceleration. It is an objective and mostly standardised method of sleep disorder evaluation, but its exact results depend on the specific hardware and algorithms that are used to quantify the level of activity of a subject. 'Usually an actigraph records both intensity and duration of movements. It is light and easy to wear and can be used in non-compliant subjects (e.g. infants, patients with dementia). The actigraph gives a recording of rest and activity over a period of weeks and can be downloaded quickly to give an instant picture when the patient comes to clinic' [98, Ch. 2]. Also, '[a]ctigraphy may be useful for studies in which PSG may

not be feasible' [14]. The main advantage of actigraphy is, therefore, the fact that it is unobtrusive and can offer a longitudinal (over many weeks) picture of the subject's physical activity. However, as Wilson and Nutt note (original emphasis) [98, Ch. 2]:

> Actigraphy may be useful for quantifying sleep as long as *wakefulness is associated with moving* and *sleep is associated with being still*. Thus the patient who lies still but awake in bed will be assumed to be asleep when the software sleep analysis is used, and people who are very restless during sleep will be assumed to have awoken, even if they have not. Actigraphy is best accompanied by some kind of daily log or diary so that unusual patterns may be seen in context.

The different methods of evaluation of sleep disorders have distinct advantages and disadvantages. Buysse *et al.* note one important advantage of sleep diaries over questionnaires and PSG recordings in the context of insomnia: 'By tracking sleep over several consecutive nights (sometimes up to 3 weeks for representative parameters such as WASO), 70 sleep diaries are more likely to capture the night-to-night variability that often characterizes the sleep of chronic insomnia. As such, sleep diaries may yield a more representative sample of an individual's sleep than 1-time questionnaires or 1 or 2 nights of PSG' [14]. Edinger and Morin state that '[t]he diagnostic value of a single night of PSG for many parasomnias is unclear, as these conditions rarely occur on a nightly basis' [30, p. 361]. Wilson and Nutt also highlight the need for large-scale longitudinal studies: 'While clinical trials will be useful in establishing cause and effect of sleep disruption on short-term health risks, more accurate measurements of sleep, in large populations followed over many years, are needed to determine whether sleep has life-long small effects on long-term health outcomes.' [64, p. 102]. Compared to sleep diaries, actigraphy offers equally longitudinal but more reliable measurements (because the measurements are objective). However, as we noted above quantifying sleep can be problematic. PSG recordings capture the entire sleep architecture of subjects, but they do not offer longitudinal measurements, and, very importantly, they can alter the sleep patterns of subjects

due to the adaptation to the environment of the laboratory, and the obstructive nature of the multiple wires that connect the different sensors to the data collection hardware. Moreover, while sleep diaries and actigraphy are cheap methods, PSG recordings are very costly [57, 14, 29] as multiple experts need to be involved for data collection and sleep staging.

If PSG recordings could be done at home, over extended periods of time, with less obstructive equipment, and less cost, they could substitute many of the measurements for which we now rely on sleep diaries, actigraphy and questionnaires. Specifically, variables 2, 3, 6, 7 and 8 from the sleep diary in Table 2.8 can be objectively recorded with PSG. Also, compared to actigraphy, a PSG recording is (for the most part) not affected by movement, so that sleep and wakefulness can be more reliably distinguished without needing to make use of the unreliable hypothesis that wakefulness is associated with moving and sleep with being still. Finally, questions regarding sleep quality and disturbances from various questionnaires could also be objectively answered with PSG recordings. We should note, however, that the subjective view of a subject regarding his or her sleep will still be important, as, for example, the diagnosis of insomnia is based on a patient's subjective complaint [14, 30].

In at-home PSG recordings there are three additional variables that need to be measured:

1. the 'Lights Off' and 'Lights On' indicators (see Section 2.3 and Table 2.4);
2. whether the subject was woken up by his or her alarm clock (See variable 4 in the sleep diary in Table 2.8);
3. whether the subject was in bed at any point in time.

An inexpensive light sensor could be used for obtaining the 'Lights Off' and 'Lights On' indicators. However, this measurement would be very noisy as electronic reading devices such as tablets and smartphones may still remain in use after subjects have switched off the lights. A mobile phone application installed in the subjects' devices can

be used in combination with the light sensor. As many people use their mobile phone as an alarm clock, a simple mobile phone application could also be used to record the alarm clock timing, but this could also be achieved with an inexpensive sound sensor. To record whether a subject was in bed at any point in time, a wrist-worn actigraph is the simplest option. Including these three variables would allow the measurement of variables 1, 4 and 5 from the sleep diary in Table 2.8, leaving only the 'Comment' field for the subject to fill in manually. Importantly, all of the variables derived from the hypnogram, shown in Table 2.4 can be calculated. Many of the questions of questionnaires for the assessment of sleep quality and disturbances could also be objectively answered.

We can therefore see that, if PSG recordings could be done at home, sleep clinicians and researchers would be able to obtain longitudinal information for a large number of variables that are essential for diagnosing and monitoring sleep disorders. We should, however, note that the information from a single night of at-home PSG would still be relatively less reliable than a PSG recording in a sleep clinic. There is a trade-off in the two methods between long-term data collection and recording accuracy.

2.5 Wearable electroencephalography

The essential sensor modality in PSG recordings is EEG, as the majority of the sleep scoring guidelines are based on EEG (see Section 2.3, and Tables 2.2 and 2.3 in particular). The other two important modalities are EOG and chin EMG. These three modalities together cover all the criteria for sleep staging. Leg EMG, respiratory variables and ECG are very important from a diagnostic perspective for many sleep disorders, but they are not used for sleep staging. Depending on the sleep disorder that one might be focusing on, they may or may not be of interest. Furthermore, for certain body movements (e.g. going out of bed) actigraphy could be used.

If PSG recordings are to be done at home, we need to examine the suitability of different modalities for long-term wearability. In terms of wearability different modalities

vary significantly in size, weight and comfort levels. A wrist-worn actigraph is possibly the most unobtrusive of all the above sensors. It is small, light, and does not cause any discomfort to a subject. ECG and leg EMG come next in terms of size, weight and comfort. The wearability of sensors that measure respiratory variables can vary depending on what is measured. Sensors that go into the nose, such as the nasal cannula, can be very obstructive. EEG, EOG and chin EMG can be quite uncomfortable when worn for a whole night.

When examining the potential wearability of different sensors one should take into account that in a sleep clinic most of these sensors are wired, so that data collection and storage are handled by a non-portable unit next to the bed [15]. In an at-home monitoring system this is not desirable. People need to be comfortable, and their movements unconstrained during the night. For example, subjects must be able to get out of bed to go to the toilet without removing the sensors.

The first step is to make the system wireless, so that there are no wires from the subject's body to a computer. Making the system wireless means that data collection has to happen on a microcontroller board attached to the subject's body, and that the data needs to be transferred wirelessly either to a local computer or to a remote server in the cloud.

Wireless communication and on-body data collection entail that there should be power provision on the body that should last for at least 10 hours (to be able to monitor the sleep of people with longer sleep duration). This means that computational requirements (on-board data handling and signal processing) and power consumption need to be minimised. The reason is that computational requirements and power consumption (which are related to each other as well) affect the overall size and weight of the board, mostly through the battery size.

There are two ways to reduce the processing and power requirements of a wearable sensor system: (1) design more energy-efficient hardware (sensors and circuits), and (2) reduce the number of sensors required by developing software that can accomplish the

task of interest using less information [15]. These two ways are frequently combined. Reducing the number of sensors not only positively affects the battery life, size and weight of a device, but also makes it easier for the subjects to wear the device without assistance, which is an important factor when we consider its wearability. Finally, if the data is transferred to the cloud this could be done via an application installed on the subject's mobile phone, so that the battery consumption on the wearable board is minimised.

The central modality in PSG is EEG. Therefore, if one had to use a single sensor for a wearable sleep monitoring system, a reasonable choice would be a single EEG channel (a single EEG channel entails two electrodes, one at the position of interest and another one as a reference). The EEG derivations that AASM recommends require either 4 electrodes (F_4-M_1, C_4-M_1 and O_2-M_1) or 5 electrodes (F_z-C_z, C_z-O_z and C_4-M_1) [49, p. 23]. In the case of the first derivation recommended by the AASM, there is a recommendation for placing backup electrodes on the opposite side of the scalp, which give the following alternative setup: F_3-M_2, C_3-M_2 and O_1-M_2 [49, p. 23]. For the second derivation the recommended backup electrodes give multiple alternative derivations by substituting F_{pz} for F_z, C_3 for C_z or C_4, O_1 for O_z, and M_2 for M_1 [49, p. 23]. Including the backup electrodes the two derivations recommended by the AASM can amount to up to 8 and 9 electrodes respectively. Some more recent PSG studies in the literature, such as [76] include as many as 21 electrodes (for 20 EEG channels).

There are two main considerations for choosing a single EEG channel. The first is the effectiveness of the channel in capturing the characteristic EEG patterns that are used by experts for sleep scoring. To that end, understanding the topographical characteristics of different frequency-band activity can be useful. For example, delta activity [32], K-complexes [40] and lower frequency sleep spindles [51] are predominantly frontal phenomena. Theta activity [32] and higher frequency sleep spindles [51] are mostly parietal phenomena. Alpha activity is predominantly an occipital phenomenon [49], but can manifest itself in frontal derivations [32].

The second consideration for choosing a single EEG channel is the stability and comfort of the electrode and reference location. Stability is particularly important since for a light wearable device there will be no backup electrodes. For longitudinal sleep monitoring comfort is a very important factor as well. In general, for the main electrode any location could be used with a cap that includes the sensor. Frontal pole (F_p) and occipital (O) locations can be very easily incorporated into a head-band, improving stability and wearability. Central (C) locations could also be added on an additional band attached to a head-band from above the two ears, or from F_{pz} to O_z (See Figure 2.1). In that case frontal (F) electrodes could be considered too.

It is typical in PSG studies for the left or right mastoid electrode (M_1 or M_2 respectively) to be used as the reference electrode. In other studies the left or right earlobe electrode (A_1 or A_2 respectively) can be also used. However, these locations can be uncomfortable or unsuitable for keeping the electrode stable. Mastoid electrodes need sticky medical tape to be stabilised. This might not be extremely uncomfortable, but can be difficult to localise easily without expert supervision. Earlobe electrodes are stabilised by using clips. These can be unstable as they might get detached when a subject moves during the night. Earlobe clips can also be uncomfortable for longitudinal sleep monitoring.

Given the wearability, stability and comfort considerations, as well as the AASM recommendations, the most promising candidate derivations for single-channel EEG are likely to be the ones derived from the F_{pz}, F_z, C_z and O_z electrodes, such as F_z-C_z, F_{pz}-C_z, and C_z-O_z.

There are two main categories of electrode technologies: traditional wet electrodes and dry electrodes [15]:

> Traditional EEG electrodes . . . are made from silver/silver chloride (Ag/AgCl) and are wet: a conductive gel is used to make the connection between the electrode and the scalp. This lowers the electrode impedance and allows recording of EEG through a high input impedance input amplifier. Over time,

> however, the conductive gel dries out and reduces the quality of the electrode contact and the EEG recording produced ... [D]ry electrodes ... require no special preparation of the subject, are simply placed on the scalp, and can be easily held in place by a hat ... Multiple potential methods have been investigated, e.g., stainless steel with various coatings for capacitive coupling of the EEG [91, 33], hybrid resistive-capacitive coupling of the EEG [67], [and] carbon nanotube and micro electro mechanical systems electrodes based on piercing the outer layers of skin for better electrical conductivity [63, 85].

There are now many consumer electronics EEG devices which utilise dry electrodes. These include Emotiv Insight (`https://emotiv.com/insight.php`), Muse (`http://www.choosemuse.com/`) and MyndPlay BrainBandXL (`http://myndplay.com/products.php?prod=9`). The major drawbacks of Emotiv Insight and Muse are the lack of stability, and their relatively low battery life (maximum 5 hours). MyndPlay BrainBandXL, on the other hand, has a battery life of 10 hours and looks very stable as it is a head-band. However, the reference electrode is on an ear clip, which can be uncomfortable for longitudinal monitoring.

There are two smartphone applications in the market that claim to be analysing sleep architecture, *without using EEG*. The first, SleepCycle (`http://www.sleepcycle.com/`) uses the accelerometer of the phone. The phone has to be placed next to the user's pillow. The output is a 'simplified' hypnogram, which includes 'Awake', 'Sleep' and 'Deep sleep' as labels in a continuous axis. As we have shown, sleep stages are discrete, but the creators of the application do not explain how they derive continuous measurements from discrete sleep staging or what their continuous scale means. Our understanding is that they merge stages N1, N2 and R under the single 'Sleep' label, which significantly lowers the validity and usefulness of the derived 'sleep architecture'. The second smartphone application is SleepRate (`https://www.sleeprate.com/`). SleepRate is the only sleep monitoring application in the consumer market for which there is an evaluation published in a scientific journal [26] (some of the co-authors of the paper work for the company that

makes the application). SleepRate uses ECG and outputs the total sleep spent in four sleep stages, and not the sleep architecture. The creators of the application have merged stages N1 and N2 into a single 'light sleep' stage.

Given the number of people that suffer from sleep disorders and the psychiatric and neurodegenerative diseases [100] that are associated with sleep, there is currently a lack of a robust, clinically validated system for longitudinal monitoring of the full sleep architecture of patients for clinical and research purposes. With the recent advances in wearable EEG, we think that it is possible to create an affordable, portable and unobtrusive sleep monitoring system for unsupervised at-home use. The core software component of such a system is a sleep scoring algorithm, which can reliably perform automatic sleep stage scoring given the subject's EEG signals.

2.6 Sleep scoring metrics

All performance metrics are derived from the *confusion matrix*. If there are N classes in a classification problem (in our case $N = 5$), the confusion matrix is an $N \times N$ matrix whose rows correspond to the true class of the training examples and whose columns correspond to the predicted class by a machine learning algorithm (or the transpose of such a matrix). Specifically, if a test example belongs to class i and it is classified as class $j \neq i$, we add 1 to the count of the (i, j) cell of the confusion matrix. If an example from class i is correctly classified as class i, we add 1 to the count of the (i, i) cell.

Using a 'raw' confusion matrix in the presence of unbalanced classes implicitly assumes that the relative importance of correctly detecting a class is directly proportional to its frequency of occurrence. This is not desirable for sleep staging. To mitigate the negative effects of unbalanced classes on classification performance measurement, what we need is effectively a normalised (per row) or 'class-balanced' confusion matrix that places equal weight into each class. This can be achieved by dividing each row of the confusion matrix with its sum (Figure 2.3). Surprisingly, in the single-channel EEG sleep staging

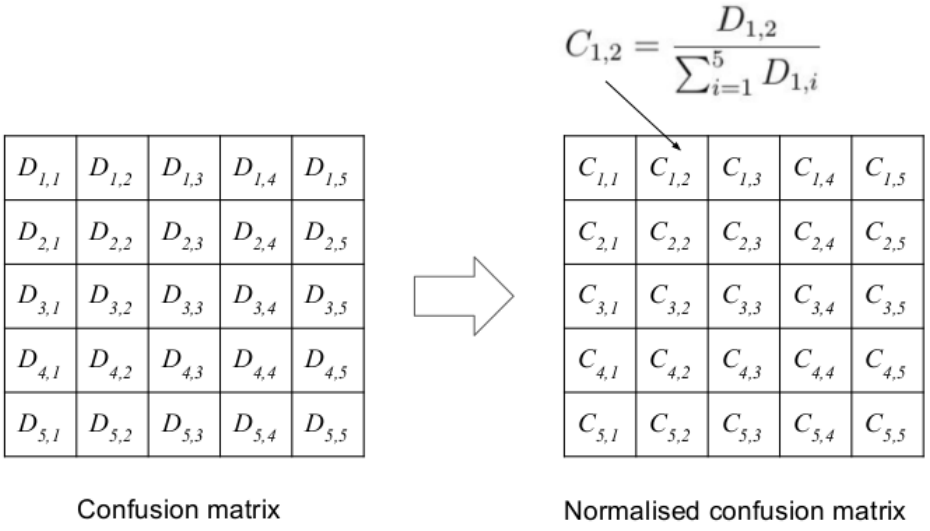

Figure 2.3: A raw confusion matrix (left) and a normalised confusion matrix (right) for a 5-class classification problem. The normalised confusion matrix is derived from the raw confusion matrix by dividing each row of with its sum.

literature there are examples of such mistakenly reported performance results using the raw confusion matrix. For this reason, we compare our work only with the studies in the literature that provide the raw confusion matrix, from which we compute the performance metrics after class-balancing.

The metrics we compute are precision, sensitivity, F_1-score, per-stage accuracy, and overall accuracy. All these metrics (apart from overall accuracy) are *binary*. However, in our case we have 5 classes. Therefore, after we perform the classification and compute the normalised confusion matrix, we convert our problem into 5 binary classification problems each time considering a single class as the 'positive' class and all other classes combined as a single 'negative' class (this is known as *one-vs-all* classification).

A normalised multiclass confusion matrix can be converted into a normalised binary confusion matrix as follows. Assume that we are doing a one-vs-all evaluation for class i. The (i, i) cell of the normalised confusion matrix is the percentage of true positives (TP). The sum of all (i, j) cells, where $j \neq i$, is the percentage of false negatives (FN), since each row of the normalised confusion matrix already sums up to 100. The sum of all (j, i) cells, where $j \neq i$, is the fraction of false positives (FP). However, in order for it

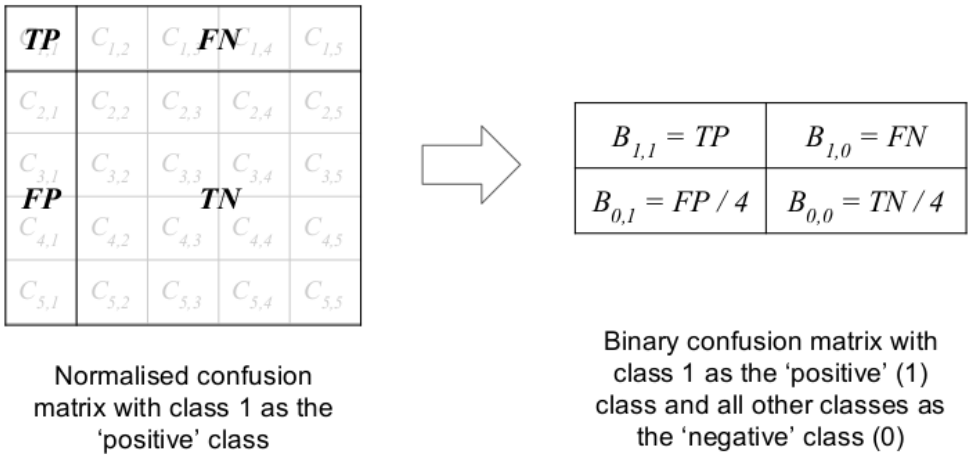

Figure 2.4: A normalised confusion matrix for a 5-class classification problem (left) divided into four quadrants by considering class 1 as the 'positive' class and all other classes as a single 'negative' class, and the binary confusion matrix derived from it (right).

to become a percentage we have to divide it by the number of all the other classes apart from class i. The sum of all the remaining cells divided by the number of classes minus one, as in the case of FP, is the percentage of true negatives (TN), since for binary performance with respect to class i misclassifications among the remaining classes are not taken into account. An illustrative example is shown in Figure 2.4.

Precision (or selectivity, or positive predictive value) is the fraction of examples classified as belonging to class i that truly belong to class i. It is computed by:

$$\text{precision} = \frac{TP}{TP + FP}. \tag{2.1}$$

Sensitivity (or recall, or hit rate) is the fraction of examples belonging to class i that were correctly classified as belonging to class i. It is computed by:

$$\text{sensitivity} = \frac{TP}{P} = \frac{TP}{TP + FN}. \tag{2.2}$$

The F_1-*score* is the harmonic mean of precision and sensitivity (recall). It is computed

by:

$$F_1\text{−score} = 2\ \frac{\text{precision} \times \text{sensitivity}}{\text{precision} + \text{sensitivity}}. \tag{2.3}$$

The F_1-score is a more comprehensive performance measure than precision and sensitivity by themselves. The reason is that precision and sensitivity can each be improved at the expense of the other. During the training of our machine learning algorithm we used F_1-score as our main learning target.

Accuracy is the number of examples correctly classified as belonging or not belonging to class i with respect to all the examples in the dataset. It is computed by:

$$\begin{aligned} \text{accuracy} &= \frac{TP + TN}{P + N} \\ &= \frac{TP + TN}{TP + FN + FN + FP}. \end{aligned} \tag{2.4}$$

Our problem is a multiclass classification problem, so we will call this accuracy per-class accuracy. The mean per-class accuracy is a measure of the class-wide accuracy of a multiclass classifier, however, accuracy is also defined for multiclass problems. Denoting the normalised confusion matrix as C for a multiclass classification problem with K classes, *overall accuracy* is computed by:

$$\text{overall accuracy} = \frac{\sum_{i=1}^{K} C_{i,i}}{\sum_{i=1}^{K} \sum_{j=1}^{K} C_{i,j}}, \tag{2.5}$$

where $C_{i,j}$ is the element of matrix C in the i-th row and the j-th column. Overall accuracy takes all misclassifications into account at the same time (i.e. they are not hidden within a 'negative' class, as in per-class accuracy), therefore it is a more suitable accuracy measure for multiclass classification problems than the mean per-class accuracy (which can nevertheless be useful if one wants to focus on particular classes).

Although PSG is considered the 'gold standard' for sleep staging, interrater agreement between human sleep scorers across subjects and stages can vary significantly. Interrater agreement is the proportion of epochs that are classified as belonging to the same sleep

stage by two or more human scorers. In [89] the consensus agreement among three experts was between 60-80%, and in [23] the authors report overall agreement of approximately 80%.

2.7 Automated sleep scoring literature review

The related work we considered in this thesis is research on automated sleep scoring using single-channel EEG. The differences between our methodology and the related literature can be identified in the following aspects: data characteristics, EEG channel used, feature extraction method, machine learning algorithm, and evaluation methodology. The high-level aspects of these studies and the present study's are outlined in Table 2.9.

The first study we considered is [34], which has the best reported performance among the existing literature. The data consists of 16 subjects (aged 30-75 years) that were admitted for possible diagnosis of sleep disorders, and the EEG channel used was C_3-A_1. The authors used time-frequency analysis based on the Choi-Williams Distribution (CWD), the Continuous Wavelet Transform (CWT), and the Hilbert-Huang Transform (HHT) and Renyi's entropy for feature extraction. The machine learning method they used was the random forest classifier. They asserted that the CWT-based feature extraction performed better, with which we compare our work in this paper. Their evaluation methodology has a serious drawback, as they train and test using epochs from all the subjects. As they state in the paper [34] 'Features from the 16 data records were assembled in one complete feature dataset ... and further divided randomly into two sets: a training dataset that contained two thirds of the features dataset ... and a testing dataset that contained the remaining one third of the features data'. This means that the training and test datasets are not independent. Moreover, in a real automated sleep monitoring system we do not expect that it will be feasible to obtain training data from each new subject that is going to use the system, because this would require them visiting a sleep clinic, which is what wearable EEG technology was proposed to avoid in the first place.

Table 2.9: The general characteristics of the single-channel EEG sleep scoring studies in the literature.

Study	**Channel**	**Data accessibility**	**Training-testing independence**	**Performance evaluation type**
Fraiwan *et al.* (2012) [34]	C_3-A_1	Private	No	Single split
Liang *et al.* (2012) [62]	C_3-A_2	Private	Yes	Single split
Liang *et al.* (2012) [62]	P_z-O_z	Open [80]	Yes	Single split
Berthomier *et al.* (2007) [11]	C_z-P_z	Private	Yes	All data

Finally, the authors of [34] evaluate their method using a single training-testing split of the data, and do not perform any type of cross-validation.

The second study we considered is [62]. In this study, the authors use two different datasets, one of which is a subset of the one we used in the present study. This offers us the opportunity for a more direct comparison. Their first dataset comprises 20 healthy subjects (aged 20-22 years), and the authors chose channel C_3-A_2. The second dataset, which is openly available in [80], comprises 8 healthy subjects (aged 21-35 years), and the authors chose channel P_z-O_z. The authors used multiscale entropy (MSE) for feature extraction from the EEG signal, and they also fitted an autoregressive (AR) model to the signal. They then trained a linear discriminant analysis (LDA) model using the MSE features and the fitted parameters of the AR model as features, employing a set of 11 *a priori* 'smoothing rules' on the hypnogram after the initial sleep scoring. MSE entropy measures the average uncertainty of a signal at progressively lower levels of granularity [22]. An AR model models each current value of a variable as a linear combination of past values of the variable. Effectively, it is a regression of the variable against itself [48, Ch. 8]. The coefficients of that regression measure the linear dependency of the current value of the variable to the corresponding past values of the variable. A LDA model is a popular linear method for classification that learns linear decision boundaries between classes with the assumption that the distributions of the data given each class are multivariate Gaussian with a common covariance matrix [41, Ch. 4]. The authors

assess the informativeness of the MSE features by visualising their correlation with the sleep stage scoring. Methodologically, this is not a correct way of assessing feature quality [1], because the class labels are revealed outside of the machine learning process, despite the fact that this is a form of training [41, pp. 241–249], and this can lead to overfitting. The authors of [62] do not state that they chose or tuned their features based on this flawed process, however, they present this correlation analysis with their final features as evidence that their features are informative for their machine learning task. With the first dataset the authors of [62] train on 10 subjects and test on the remaining 10 subjects. With the second dataset the authors train on 4 subjects and test on the remaining 4 subjects. Finally, the authors evaluate their method using a single training-testing split of the data, and do not perform any type of cross-validation.

The third study we considered is [11]. In this study, the authors use a dataset comprising 15 subjects (aged 29.2 $\pm$ 8 years, 9 female and 6 male). Unfortunately, the feature extraction methods and the machine learning algorithm are not described in detail in [11]. The general information that the authors provide is that they use spectral/temporal feature extraction with microstructure detection, rough identification of awakening and REM, and 'an adaptive fuzzy logic iterative system to repeatedly update the sleep stage pattern definitions' [11] as their machine learning algorithm. The machine learning algorithm is already trained and is tested on all 15 subjects. No information on training is provided by the authors.

We did not include the study in [18] because the authors do not report the confusion matrix.

There are two main limitations in the existing literature. First, regarding the results of the proposed methods, in all three studies we observe imbalance in the scoring performance across sleep stages. For example, the F_1-score for the worst-classified sleep stage (N1) can be as low as 30% in [62]. Second, regarding the evaluation methodology, in all three studies the authors evaluated their methods using a single training-testing split of the data, and did not perform any type of cross-validation. Furthermore, in [34] the

authors trained and tested their algorithm using epochs from all subjects, which means that the training and testing datasets were not independent. In our work we mitigated skewed sleep scoring performance in favour of the most represented sleep stages, and addressed the problem of misclassification errors due to class imbalance in the training data while significantly improving worst-stage classification. Our experimental design employs cross-validation across subjects, ensuring independence of training and testing data.

2.8 Data: The PhysioNet dataset

The dataset that we used to evaluate our methods is a publicly available sleep PSG dataset [54] available on the PhysioNet repository [37] that can be downloaded from [79]. We used the first dataset available on [79], which was originally recorded between 1987-1991 for a study of age effects on sleep in healthy subjects [70]. The data was collected from electrodes F_{pz}-C_z and P_z-O_z. The sleep stages were scored according to the Rechtschaffen and Kales guidelines [81]. The epochs of each recording were scored by a single expert (6 experts in total). The sleep stages that are scored in this dataset are Wake (W), REM (R), Non-REM stages 1–4 (N1, N2, N3, N4), movement and not scored. For the work in this thesis, we removed the very small number of movement and not scored epochs (not scored epochs were at the start or end of each recording), and also merged the N3 and N4 stages into a single N3 stage, as it is currently the recommended by the American Academy of Sleep Medicine (AASM) [49, 87]. There were 61 movement epochs in our data in total, and only 17 of the 39 recordings had movement artifacts. The maximum number of movement epochs per recording was 12. The rationale behind the decision of removing the movement epochs was based on two facts. First, these epochs had not been scored by the human expert as belonging to any of the five sleep stages, as it is recommended in the current AASM manual [49, p. 31]. Second, their number was so small that they could not be used as a separate 'movement class' for learning. The public dataset includes 20 healthy subjects, 10 male and 10 female, aged 25–34 years. There are

two approximately 20-hour recordings per subject, apart from a single subject for whom there is only a single recording. To evaluate our method we used the in-bed part of the recordings. The sampling rate is 100 Hz and the epoch duration is 30 seconds.

Chapter 3

Signal processing background

Signal decomposition is the process of analysing a signal by separating it into more primitive components. The aim of performing signal decomposition is to use those primitive components to better understand a given signal. The Fourier transform, wavelet filtering, bandpass filtering and the Hilbert transform are examples of signal decomposition methods that are commonly used in practice. In machine learning practice in particular, signal decomposition is commonly used as a feature extraction methodology, in which the primitive components and their relationships are the features that are used as input to a machine learning algorithm. In this chapter our exposition will be based on real-valued one-dimensional signals, and we will focus on wavelet-based signal decomposition (or wavelet filtering) using complex Morlet wavelets. This exposition provides the necessary background for understanding the connection between deep neural networks (particularly convolutional neural networks) and signal decomposition methods.

3.1 The dot product

The basis for signal decomposition using wavelets is convolution. In turn, the basis of convolution is the dot product. The dot product $\langle \mathbf{a}, \mathbf{b} \rangle$ between two real column vectors $\mathbf{a}$ and $\mathbf{b}$ of equal length M is defined as the sum of the element-wise multiplication of the

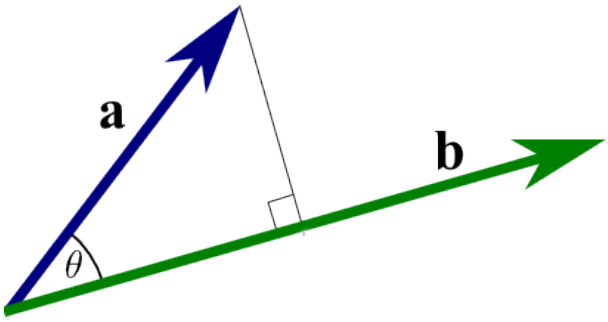

Figure 3.1: Two vectors **a** and **b** and the angle θ between them.

two vectors:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{M} a_i b_i = \mathbf{a}^T \mathbf{b}, \tag{3.1}$$

where a_i is the i-th element of vector **a**. The dot product is a measure of similarity between two signals. The geometric intuition behind the dot product is that it is a measure of how much two vectors point in the same direction. It is the product of the length of the 'part' of the vectors that points in the same direction. It can be written as:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos\theta_{\mathbf{a},\mathbf{b}}, \tag{3.2}$$

where θ is the angle between the vectors **a** and **b** (Figure 3.1), and $\|\mathbf{a}\|_2$ is the l^2-norm or Euclidean norm of vector **a**:

$$\|\mathbf{a}\|_2 = \sqrt{\sum_{i=1}^{M} |a_i|^2}. \tag{3.3}$$

The term $\|\mathbf{a}\|_2 \cos\theta_{\mathbf{a},\mathbf{b}}$ in Equation 3.2 is the length of the projection of vector **a** on vector **b**. From Equation 3.2, the cosine of the angle θ can be written as:

$$\cos\theta_{\mathbf{a},\mathbf{b}} = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}. \tag{3.4}$$

If the two vectors are normalised by subtracting their means from each of their elements $(a_i - \bar{\mathbf{a}})$ the cosine of the angle between the two vectors is equivalent to the Pearson correlation coefficient between these two vectors. The cosine is in the range $[-1, 1]$. If the

dot product is 0, the cosine of the angle between the two vectors is 90° (or, equivalently, the length of the projection of one vector onto the other is 0), and therefore the two vectors are orthogonal to each other. Using the dot product as a measure of similarity implies that two vectors are maximally dissimilar if they are orthogonal to each other ($\cos\theta = 0$), and maximally similar if they are collinear and pointing to the same direction ($\cos\theta = 1$). If $\cos\theta = -1$ then the two vectors are still collinear but point towards exactly opposite directions.

3.2 Convolution

Convolution can be intuitively viewed as the result of a sliding dot product operation. Our case of interest is when one of the signals, for example $\mathbf{b}$ with length N, is shorter than signal $\mathbf{a}$ with length M, i.e. $N < M$. We could then slide signal $\mathbf{b}$ over signal $\mathbf{a}$, and compute the dot product $M - N + 1$ times. The sliding dot product is given by:

$$(\mathbf{a} \star \mathbf{b})_t = \sum_{i=0}^{N-1} a_{t+i} b_{i+1}, \text{ for } 1 \leq t \leq (M - N + 1). \tag{3.5}$$

In signal processing Equation 3.5 is called the *cross-correlation* between signals $\mathbf{a}$ and $\mathbf{b}$. If signal $\mathbf{b}$ is flipped (time-reversed) before this operation, this is the definition of *convolution*, which is denoted as $\mathbf{a} * \mathbf{b}$. Formally, the element of the convolution $(\mathbf{a} * \mathbf{b})_t$ at time t is defined as:

$$(\mathbf{a} * \mathbf{b})_t = \sum_{i=0}^{N-1} a_{t+i} b_{N-i}, \text{ for } 1 \leq t \leq (M - N + 1). \tag{3.6}$$

The subscript $N - i$ for vector $\mathbf{b}$ is the flipping operation. If vector $\mathbf{b}$ is symmetric, convolution and cross-correlation produce the same result. Equation 3.6 is time-domain convolution. Equation 3.6 does not permit signal $\mathbf{b}$ to fall outside the signal $\mathbf{a}$. The result of the convolution has length equal to $M - N + 1$ (commonly called 'valid' in signal processing frameworks). In another form of convolution, signal $\mathbf{a}$ is padded with $N - 1$

zeros at its start and $N-1$ zeros at its end. The result of this convolution has length equal to $M+N-1$ ('full'). A third type of convolution is to retain only the central part of a full convolution that has length equal to vector $\mathbf{a}$. This is done by removing $N/2$ points from the start of the result and $N/2+1$ points from the end of the result, so that the convolution has length M ('same'). Both valid-length and same-length convolution can be extracted from the result of full-length convolution.

Convolution can also be computed in the frequency domain. Denoting the Fourier transform of $\mathbf{a}$ as $\mathcal{F}[\mathbf{a}]$ and the inverse Fourier transform as $\mathcal{F}^{-1}[\mathbf{a}]$, from the Convolution Theorem we have [96]:

$$\mathcal{F}[\mathbf{a} * \mathbf{b}] = \mathcal{F}[\mathbf{a}] \cdot \mathcal{F}[\mathbf{b}] \tag{3.7}$$

$$\mathbf{a} * \mathbf{b} = \mathcal{F}^{-1}\big[\mathcal{F}[\mathbf{a}] \cdot \mathcal{F}[\mathbf{b}]\big], \tag{3.8}$$

where $\cdot$ denotes element-wise multiplication. Equation 3.8 states that convolution in the time domain is multiplication in the frequency domain. To apply Equation 3.8 to signals $\mathbf{a}$ and $\mathbf{b}$, signal $\mathbf{a}$ needs to be padded with $N-1$ (the length of signal $\mathbf{b}$ minus 1) zeros in its end, and signal $\mathbf{b}$ needs to be padded with $M-1$ (the length of signal $\mathbf{a}$ minus 1) zeros in its end. The result is full-length convolution, of length $M+N-1$. From that valid-length or same-length convolution can be extracted as needed. Frequency-domain convolution can be computed more efficiently than time-domain convolution by using the Fast Fourier Transform (FFT) algorithm.

3.3 Time-frequency analysis

Time-frequency analysis is the process of decomposing a signal into different components by designing suitable filters (or wavelets, or kernels), with which the signal is convolved. Sliding the (flipped) filter across the signal and taking the dot product between the two we effectively compute the similarity between the signal and the filter over time (in

the case of images it would be similarity across space, using a two-dimensional filter). Convolution can be interpreted as a measure of how similar the signal is to the filter at different locations within the signal. Alternatively, we can interpret it as a filtering operation in the sense that only the parts of the signal that are similar to the filter will 'pass' from it (the more similar a part of the signal is to the filter the further away its dot product with the filter is from zero).

When wavelet filtering is applied to signals whose dimension is time, like EEG signals, it is commonly called time-frequency analysis. As wavelets of different frequency content are passed across a signal, at each point in time the dot product of the signal with the (flipped) wavelet is the similarity of the signal with the wavelet. When the convolution is complete the result is similarity of the signal with the wavelet across time, or the amount of the frequency content of the wavelet which is present in the signal. Time-frequency analysis is the analysis of that resulting signal.

Compared with the Fourier transform, time-frequency-based feature extraction has the advantage of extracting features that capture the mixture of frequencies and their interrelations at different points in time. These interrelations can be useful features for problems like sleep stage scoring (see Tables 2.2 and 2.3).

The time-frequency analysis method we will analyse is complex Morlet wavelets, a widely used wavelet-based time-frequency analysis method (see, for example, Chapters 12 and 13, pp. 141–174 in [20]). Complex Morlet wavelets are very intuitive for describing time-frequency analysis, as a complex Morlet wavelet is a complex sine wave windowed with a Gaussian taper, and its peak frequency is simply the frequency of the sine wave. By virtue of those characteristics, when they are used to filter other signals the interpretation of filtering in terms of the frequency content of those signals is also intuitive. The equation of a complex Morlet wavelet $\mathbf{w}$ at time t, given amplitude a, peak frequency f and standard deviation s, is:

$$w_t = a \exp\left(-\frac{t^2}{2s^2}\right) \exp\left(i2\pi f t\right), \tag{3.9}$$

where the first exponential is a Gaussian and the second exponential is a complex sine wave, with f the peak frequency of the wavelet in Hz, t the timepoint in seconds, s the standard deviation of the Gaussian taper in seconds, and a the amplitude of the Gaussian taper (there exist alternative parameterisations of the equation of a complex Morlet wavelet).

The reciprocal of the lowest peak frequency of the wavelets (seconds per cycle) cannot be larger than the duration of the signal in seconds. For example, if the peak frequency is 2 Hz we must have a signal of at least $1 \div 2 = 0.5$ seconds in duration. Moreover, it is better to have multiple cycles of the wavelet per epoch, commonly at least 4 [20]. In the example above, it would be better to have a signal of $4 \times 0.5 = 2$ seconds in duration. The highest peak frequency of the wavelets cannot be higher than the Nyquist frequency, i.e. one half of the sampling rate. For example, for a signal sampled at 100 Hz, the highest peak frequency of the wavelets can be at most 50 Hz.

The standard deviation s is given by the equation:

$$s = \frac{n}{2\pi f}, \tag{3.10}$$

where n is the number of wavelet cycles, which define the width of the wavelet. This parameter controls the trade-off between temporal and frequency precision. Specifically, increasing the number of cycles increases the frequency precision but decreases the temporal precision, while decreasing the number of cycles increases the temporal precision.

The amplitude a of the Gaussian taper is given by the equation:

$$a = \left(s^2\pi\right)^{-1/4}. \tag{3.11}$$

For time-frequency analysis using complex Morlet wavelets there are therefore two sets of parameters that need to be decided, the peak frequencies f and the number of wavelet cycles n per frequency.

After creating a complex Morlet wavelet we take its convolution with our signal. The

power at the frequency band centred at the peak frequency of the wavelet is the squared length of the result of the convolution in the complex plane (the result of convolution of a real signal with a complex wavelet is a complex signal). For a complex number $a + ib$ the length in the complex plane $|a + ib|$ is defined as $\sqrt{a^2 + b^2}$. The square of the length is therefore $a^2 + b^2$. This is most easily computed by taking the element-wise product of the convolution with its complex conjugate. The complex conjugate of a complex number $a + ib$ is defined as $a - ib$. Multiplying the two yields $a^2 + b^2$ [20, pp. 158-162].

Chapter 4

Deep neural networks

In this chapter we give an overview of deep neural networks (DNNs) [7, 59]. We aim to provide a detailed view of DNNs, following an exposition that will set the scene for convolutional neural networks and with a view to highlighting the links between the signal processing background of Chapter 3 and neural networks. We begin by defining supervised learning (classification and regression) and unsupervised learning. Sleep stage scoring is a classification problem, and, therefore, our exposition concentrates on this type of problem. The machine learning methods we describe in this chapter are logistic and softmax regression, 'standard' neural networks, autoencoders, and, finally, the method we used for our work in Chapter 6, stacked sparse autoencoders. We give the details of prediction (forward propagation), optimisation (stochastic gradient descent), and partial derivative computation (backpropagation) in neural networks. We also describe regularisation as a method for avoiding overfitting.

4.1 Machine learning background

Machine learning is the field that concerns itself with the construction of models that learn how to predict or infer the value or category of a variable of interest (the *target variable* or *label*) given the values of another set of variables that comprise a data point or *input vector*, or to discover patterns in unlabelled data, i.e. using only the input vectors.

When labelled data are available, the machine learning paradigm is called *supervised learning*. When there are no labels in the data the machine learning paradigm is called *unsupervised learning*. In unsupervised learning the objective is to find patterns and structure in the data. The most common form of unsupervised learning is clustering. Other machine learning paradigms, which are beyond the scope of this thesis, are *semi-supervised learning* (combining both labelled and unlabelled data), *reinforcement learning* (learning based on rewards to actions), and *transfer learning* (using models learned in similar tasks). In this thesis we focus on supervised learning for classification, and, to a lesser extent, unsupervised learning.

The statistical models used in machine learning have parameters that are learned by using training data. Models may also contain hyperparameters which are pre-set. After learning, the model with the learned parameters is used for prediction or inference in previously unseen data points, the testing data. The generalisation performance (on the testing data) of a machine learning model is what we are interested in. Good performance on training data alone is not an indication of a good model. In particular, if a model exhibits good performance on the training data, but poor performance on testing data, this is called *overfitting*.

4.1.1 Supervised learning

In classification problems the aim is to associate an input vector with one or more classes to which it belongs. An input vector can belong to more than one class if the classes are not mutually exclusive. In this thesis we will focus on classification problems with mutually exclusive classes. For example, the input vector may comprise multiple variables from a patient record (e.g. age, gender, blood pressure, triglyceride level, cholesterol level), and the target variable is whether the patient belongs to one or more different classes (e.g. healthy or with cardiovascular problems). In regression problems the aim is to associate an input vector with a continuous-valued vector (or scalar). In our previous example, the target variable could be the patient's blood glucose level. Sleep stage scoring

is a classification problem. Our exposition of deep neural networks and convolutional neural networks will therefore be focused on classification.

4.1.2 Unsupervised learning

In unsupervised learning class labels are not available. Common objectives in unsupervised learning are the generation of classes from the data by clustering together input vectors which are statistically similar, learning a model of 'normality' and classifying abnormal input vectors based on their dissimilarity with that model, or learning a more parsimonious representation of the input data, e.g. by reducing their dimensionality or imposing sparsity constraints. In this thesis we will use a type of unsupervised neural network model, the sparse autoencoder, which can be used to extract features from a dataset by imposing sparsity constraints.

4.1.3 Cross-validation

To evaluate the generalisability of a model, we need to validate it on a separate dataset with previously unseen data. It is common to split the full dataset into a training and testing set with, e.g., 80% of the data in the training dataset and 20% of the data in the testing dataset.

The problem with this approach approach is that it sacrifices the data in the testing dataset. There are applications for which collecting and labelling the data is expensive, and, in such cases, it would be desirable that the held-out testing dataset be used. Moreover, the particular split between the training and the testing dataset might not be representative. It is also impossible to assess the variability in performance. All of these issues are addressed by using *cross-validation*. The most commonly used method of cross-validation is *K-fold cross-validation*. In K-fold cross-validation the dataset is split into K folds, and the training-testing process is repeated K times with one of the K folds as the testing dataset and the rest as the training dataset each time. The reported performance is commonly the mean performance across the K folds, which is a better

estimate of out-of-sample performance than a single training-testing split of the data. In most applications K-fold cross-validation is applied over data points. However, in applications such as sleep stage scoring it is better to split the data in terms of subjects, because of the inherent similarities between the data points of the same subject. Doing K-fold cross-validation at the level of data points when the data is obtained from different subjects may erroneously inflate the performance of a machine learning algorithm.

The training dataset (for each fold in the case of K-fold cross-validation) can be further split into a training dataset and a validation dataset. The validation dataset is used to assess the performance of the model before finally testing it on the testing dataset. This is useful for two main reasons. The first is that complex models, such as DNNs, have non-trainable *hyperparemeters*. To evaluate their choice an independent validation dataset is needed, before fixing them and testing on the testing dataset. The second reason is that many models, among them DNNs, are trained iteratively, and a stopping criterion is needed to avoid overfitting. A common choice for the stopping criterion is the performance of the model on the validation dataset, after each training iteration. If performance keeps on improving on the training dataset, but does not improve or deteriorates on the validation dataset, this is an indication of overfitting, and training should stop.

4.1.4 Overfitting

Overfitting is the situation when a model exhibits good performance on the training dataset but poor performance on the testing dataset. There are two main reasons for overfitting. The first is model complexity. The more complex a model is the better it can model the training data. However, fitting well to training data may lead to over-specialisation to the training data, so that the model is not general enough to capture relationships out of sample. A common solution to that is reducing model complexity. Model complexity can be reduced either by reducing the number of trainable parameters in a model, or by imposing constraints on the magnitude that these parameters can take.

A common way to restrict the magnitude of the parameters is regularisation, which we describe in Section 4.9.

4.2 Logistic regression

Logistic regression is the most widely-used supervised learning algorithm for classification in binary (two-class) problems. The logistic regression algorithm outputs a class probability per class, which can then be combined with a decision rule (most commonly arg max) to output the estimated class to which the input belongs. Importantly, for our purposes, logistic regression is also the basis of neural networks. We will review logistic regression in some detail as it is fundamental for understanding the mechanics of neural networks.

Logistic regression can be viewed as the minimal neural network. Visualising and analysing logistic regression as the minimal neural network will help us assemble a full-scale neural network by combining multiple of those minimal neural networks. For this purpose, in what follows we will use the terminology of neural networks to describe logistic regression. The minimal neural network is the basic building block for doing classification using neural networks, and all neural network variants, even the most complex, are extensions of it. The minimal neural network, shown in Figure 4.1, has only two layers: an input layer (blue circles), and an output layer with a single unit (red circle).

Let us introduce our notation. For a binary classification problem with real-valued input the input comes in the form of a vector $\mathbf{x}$ in $\mathbb{R}^N$. Each input vector is associated with a binary categorical variable y, for which, for notational convenience, we arbitrarily assign the numbers 0 and 1 to its respective possible values (the two classes), so that $y \in \{0, 1\}$. By convention, for example, in many biomedical problems 0 usually denotes healthy individuals, and 1 individuals that suffer from a particular condition.

The output layer of logistic regression outputs the probability that the class associated with the input is the first one, formally that $p(y = 0|\mathbf{x})$, as shown in Figure 4.1. The

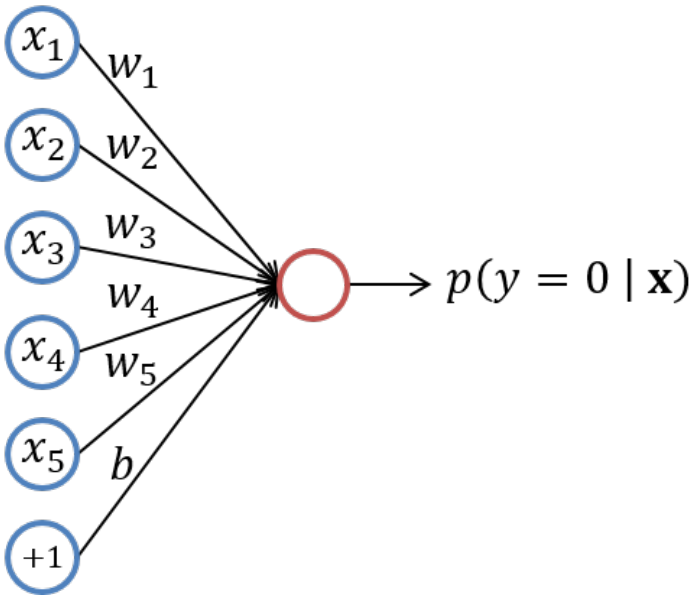

Figure 4.1: Logistic regression, or the minimal neural network

probability that the input belongs to the other class is then simply given by: $p(y = 1|\mathbf{x})$ = 1 - $p(y = 0|\mathbf{x})$, assuming that the classes are exhaustive (otherwise we would need a third 'undefined' class, and our problem would no longer be binary).

In Figure 4.1, there is an arrow from each element of the input vector $\mathbf{x}$ to the unit of the output layer. Each of these arrows is associated with a weight, so that the i-th element x_i of vector $\mathbf{x}$ is associated with a weight w_i. For each (w_i, x_i) pair the unit performs a multiplication. In the input we also include a *bias element* which is always equal to 1. This bias element is associated (i.e. multiplied) with the *bias weight* b.

To keep the notation uncluttered, without loss of generality, we will assume that the input vector $\mathbf{x}$ includes the bias element and that the weight vector $\mathbf{w}$ includes the bias weight b. In the remainder of the thesis we will not explicitly refer to the bias elements or weights. N will then be the number of elements in the input vector *including* the bias element.

In the unit of the output layer the results of the multiplications are added up. The result of the addition is then supplied as input to the logistic function (hence the name logistic regression). In the machine learning literature the logistic function is commonly called the *sigmoid* function, although there are many other types of sigmoid functions.

We will use the term sigmoid for the logistic function in this thesis.

Denoting the sigmoid function as f, the computation that is performed inside the output unit is:

$$\begin{aligned} a(\mathbf{x}; \mathbf{w}) &= f(w_1 x_1 + w_2 x_2 + \cdots + w_N x_N) \\ &= f\Big(\sum_{i=1}^{N} w_i x_i\Big). \end{aligned} \tag{4.1}$$

In neural network terminology, the function f is called an *activation function*. In logistic regression the activation function is by definition the sigmoid function, which is given by the equation:

$$f(z) = \frac{1}{1 + e^{-z}}. \tag{4.2}$$

In neural network terminology, the result of the computation in Equation 4.1 is called the *activation* of the output unit. We observe that Equation 4.1 can be more compactly written using the definition of the dot product from Equation 3.1:

$$\begin{aligned} a(\mathbf{x}; \mathbf{w}) &= f(\mathbf{w}^T \mathbf{x}) \\ &= f(\langle \mathbf{w}, \mathbf{x} \rangle). \end{aligned} \tag{4.3}$$

This observation is fundamental for connecting wavelet filtering with convolutional neural networks, as we will show in Chapter 5.

The logistic function outputs values in the $[0, 1]$ interval, and is therefore suitable as an estimate for a probability. This means we can view the activation of the output unit given in Equation 4.1 as the probability that the class associated with the input is the first one, i.e. that $p(y = 0|\mathbf{x})$, as shown in Figure 4.1. The probability $p(y = 0|\mathbf{x})$ is then given by:

$$p(y = 0|\mathbf{x}) = \frac{1}{1 + \exp[-(\mathbf{w}^T \mathbf{x})]}, \tag{4.4}$$

where $\exp(\cdot)$ is the exponential function.

The logistic regression parameters $\mathbf{w}$ can be learned by minimising its cost function with respect to $\mathbf{w}$ over a set of K training examples. Each training example is a pair of an input vector and its associated label, identified by a superscript. Concretely, the i-th training example is denoted as $(\mathbf{x}^{(i)}, y^{(i)})$. The logistic regression cost function (commonly called *log loss*) is:

$$J(\mathbf{w}; \mathbf{x}) = -\frac{1}{K}\left[\sum_{i=1}^{K} y^{(i)} \log p(y = 0|\mathbf{x}^{(i)}; \mathbf{w}) + (1 - y^{(i)}) \log p(y = 1|\mathbf{x}^{(i)}; \mathbf{w})\right]. \quad (4.5)$$

The parameters of logistic regression are learned by minimising Equation 4.5 with respect to them.

4.3 Softmax regression

If there are more than two classes in the data, we use the generalisation of logistic regression called multinomial logistic regression or *softmax* regression. In softmax regression there is a different weight vector $\mathbf{w}_j$ associated with each class j. For C classes, the activation function for each class c becomes:

$$f(z_c) = \frac{e^{-z_c}}{\sum_{d=1}^{C} e^{-z_d}}. \quad (4.6)$$

The class probability for each class j is then given by:

$$p(y = j|\mathbf{x}; \mathbf{w}_j) = \frac{\exp[-(\mathbf{w}_j^T \mathbf{x})]}{\sum_{d=1}^{C} \exp[-(\mathbf{w}_d^T \mathbf{x})]}. \quad (4.7)$$

The cost function then becomes:

$$J(\mathbf{w}_1, \ldots, \mathbf{w}_C) = -\frac{1}{K}\left[\sum_{i=1}^{K}\sum_{d=1}^{C} \mathbf{1}\{y^{(i)} = j\} \log p(y = j|\mathbf{x}^{(i)}; \mathbf{w}_j)\right], \quad (4.8)$$

where $\mathbf{1}\{\cdot\}$ is the indicator function (returns 1 if the condition is satisfied and 0 otherwise).

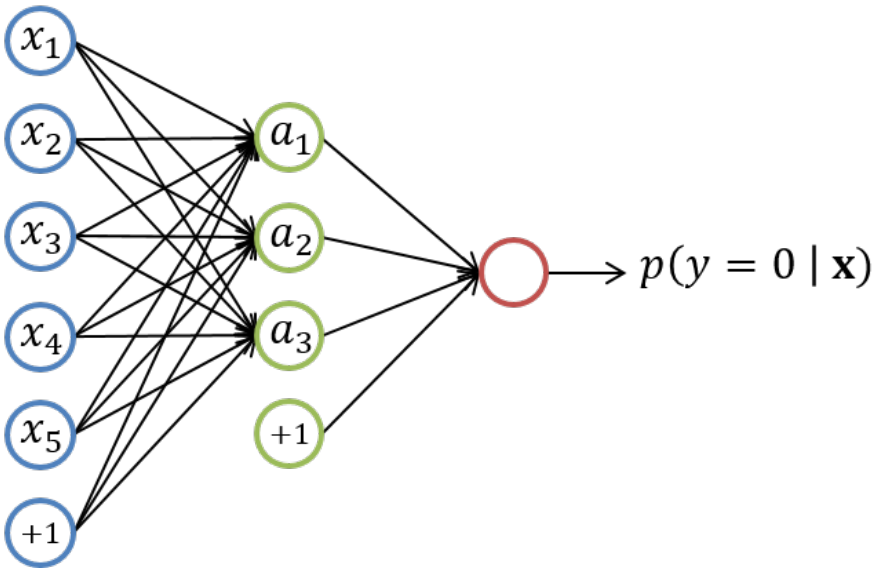

Figure 4.2: A single-layer neural network

4.4 Single-layer neural networks

Going from logistic regression to a neural network is as straightforward as adding an intermediate *hidden layer* between the input and the output layers of logistic regression (Figure 4.2). A hidden layer is composed of multiple units which perform the same computation as the output unit, as given in Equation 4.1. We should note that the weights associated with each unit of the hidden layer are different.

The only difference from logistic regression is that the activation function f of the units in the hidden layer can take different forms different from the sigmoid function (which is still one of the most common choices). We will not describe all the possible activation functions in this thesis, but it is worth mentioning the two other commonly used activation functions.

The first commonly used activation function is the *rectifier* function, which is defined as:

$$f(z) = \max(0, z). \tag{4.9}$$

A unit that employs the rectifier activation function is commonly called a rectified linear unit (ReLU) in the literature.

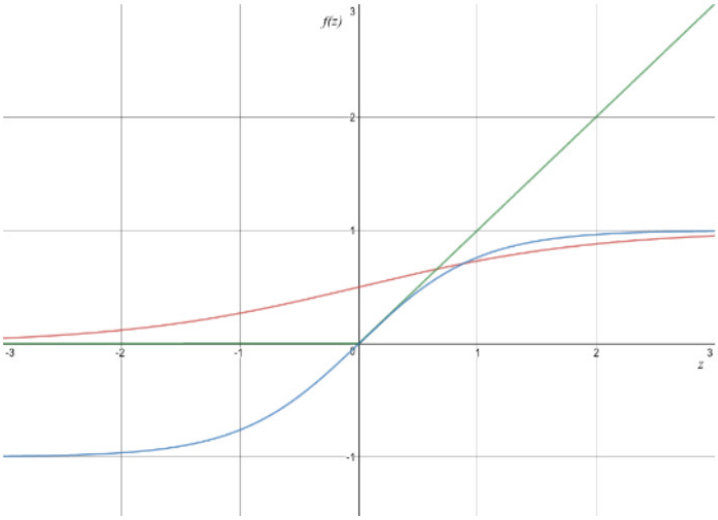

Figure 4.3: Activation functions - green: ReLU, red: sigmoid, blue: tanh

The second commonly used activation function is the hyperbolic tangent (its mathematical notation is tanh), which is another form of sigmoid function. It is defined as:

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{4.10}$$

The sigmoid, tanh, and rectifier activation functions are shown in Figure 4.3.

Each unit of the hidden layer is connected with each of the elements in the input layer, and produces a single activation (Equation 4.3). The set of all the activations of the hidden layer then becomes the input of the output layer, as shown in Figure 4.2.

We remind that the output layer is simply logistic regression. The activations of the hidden layer can then be viewed as features that are used for logistic regression instead of the input itself. The key is that all the parameters (weights) of the neural network that are responsible for producing those features are learned based on the error of the logistic regression classifier in the output layer of the neural network. We describe the learning algorithm in a later section, after our exposition of deep neural networks, which comes next.

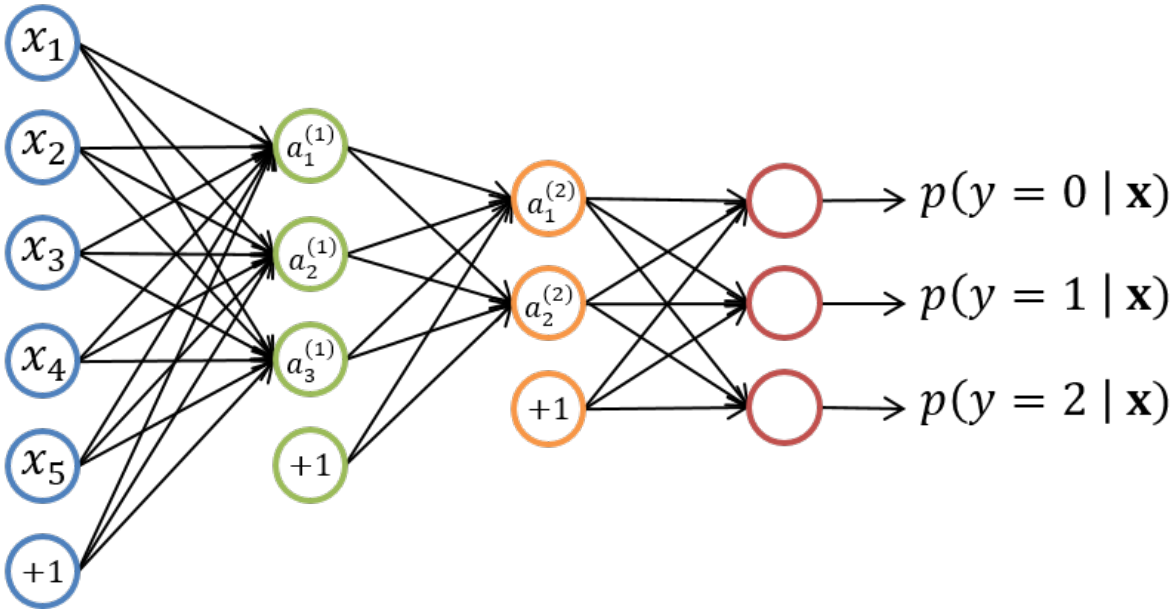

Figure 4.4: An example neural network with 2 hidden layers and 3 classes

4.5 Deep neural networks

The final step that is needed to create a deep neural network is to add more hidden layers to it, as shown in Figure 4.4. In a deep neural network the activations of all the units of a hidden layer become the input for the next (hidden or output) layer. The mechanics of a deep neural network is exactly the same as in the single-layer neural network. A deep neural network of this kind is sometimes called a *multilayer perceptron* [84] in the literature.

We show an example of a deep neural network in Figure 4.4. This network has two hidden layers with 4 and 3 units respectively, and it is applied to a 3-class classification problem. When the classification problem has more than two classes the output layer is softmax regression instead of logistic regression. The number of layers and units in a deep neural network is called the *architecture* of the network. A deep neural network is simply a neural network with a large number hidden layers.

LeCun, Bengio and Hinton provide us with the intuition of why making a neural network deep works [59]: 'Deep neural networks exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by

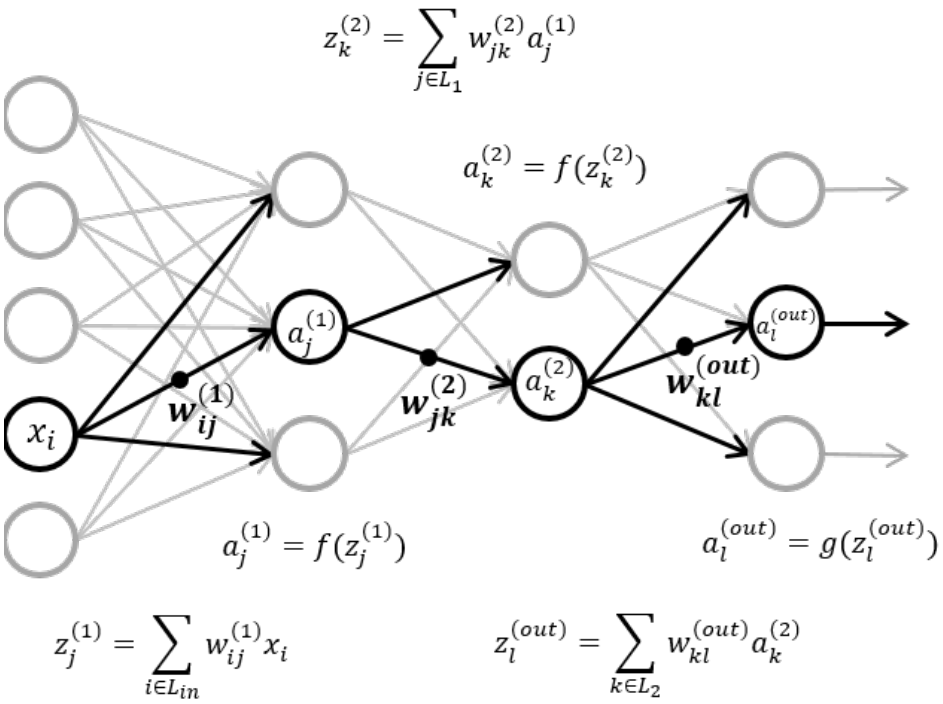

Figure 4.5: Forward propagation

composing lower-level ones.' Effectively, the activations of each consecutive hidden layer can be seen as hierarchically composed features. The activations of the first hidden layer are 'first-order' features, the activations of the second hidden layer are features of the first-order features or 'second-order' features, and so on.

4.6 Prediction in deep neural networks: Forward propagation

Let us assume that the parameters (weights) of the neural network in Figure 4.4 have been learned with a use of a training dataset. This means that the network can now be used for predicting the class labels of new, previously unseen inputs. To predict the class label associated with a training example we apply consecutively Equation 4.3 starting from the input layer. This process is called *forward propagation* because the activations of each layer are propagated *forward* to the next layer of the network.

To better understand forward propagation we introduce some notation. Here we follow the exposition in [59]. Consider the 2-layer neural network shown in in Figure 4.5. Let x_i be the i-th element of the input, $a_j^{(1)}$ be the activation of the j-th hidden unit of the first hidden layer (the layer number is denoted with the superscript (1)), $a_k^{(2)}$ be the activation of the k-th unit of the second hidden layer, and $a_l^{(out)}$ be the activation of the l-th unit of the output layer. Accordingly the weight that connects the i-th element of the input with the j-th unit of the first hidden layer is denoted as $w_{ij}^{(1)}$, the weight that connects the j-th unit of the first hidden layer with the k-th unit of the second hidden layer is denoted as $w_{jk}^{(2)}$, and, finally, the weight that connects the k-th unit of the second hidden layer with the l-th unit of the output layer is denoted as $w_{kl}^{(out)}$. The set of unit indexes in each layer is denoted as L_{layer}, so that the subscript is the layer that we refer to. Finally, f is the activation function of the hidden layers, and g is the activation function of the output layer. As we have already mentioned f and g may or may not be the same.

Taking the neural network in Figure 4.5 as our example, forward propagation proceeds in the following steps:

$$z_j^{(1)} = \sum_{i \in L_{in}} w_{ij}^{(1)} x_i \tag{4.11}$$

$$a_j^{(1)} = f\big(z_j^{(1)}\big) \tag{4.12}$$

$$z_k^{(2)} = \sum_{j \in L_1} w_{jk}^{(2)} a_j^{(1)} \tag{4.13}$$

$$a_k^{(2)} = f\big(z_k^{(2)}\big) \tag{4.14}$$

$$z_l^{(out)} = \sum_{k \in L_2} w_{kl}^{(out)} a_k^{(2)} \tag{4.15}$$

$$a_l^{(out)} = g\big(z_l^{out}\big) = p(y = l|\mathbf{x}) \tag{4.16}$$

The input Equations 4.11 and 4.12 and the output Equation 4.15 and 4.16 are unique in any neural network. If a network has H hidden layers there are going to be $H - 1$ hidden layer equation pairs like Equations 4.13 and 4.14.

As we indicate in Equation 4.16 (with a little abuse of notation) the activation of the l-th unit of the output layer is equal to the probability that the class associated with the input is l. The simplest and most common way to predict a class label using these class probabilities is by choosing the class that has the maximum probability:

$$y = \arg\max_{l} p(y = l|\mathbf{x}). \tag{4.17}$$

In the following subsection we explain how the weights that are used to get the class labels from forward propagation are learned.

4.7 Learning in deep neural networks: Stochastic gradient descent

To learn the optimal weights for the neural network, we need to minimise the cost function of logistic regression (Equation 4.5) or softmax regression (Equation 4.8) depending on whether our classification problem is binary or multi-class. Since logistic regression is just a special case of softmax regression we will focus on softmax regression for the remainder of this section.

Minimising the softmax cost function with respect to the weights in all the layers of the neural network has no closed form solution. For this reason, iterative optimisation algorithms are used to learn the weights of neural networks.

The most widely used iterative optimisation algorithm in neural networks is *stochastic gradient descent*. We will begin with the description of simple gradient descent, and then will proceed with describing stochastic gradient descent.

Let us denote the set of all the weights of a neural network as W. In its most basic form one iteration of gradient descent updates each weight $w_{ij}^{(l)}$ from the i-th unit of the $(l-1)$-th layer (if $l-1=0$ then it is the input layer) to the j-th unit of the l-th layer

in the following way:

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \alpha \frac{\partial J(W)}{\partial w_{ij}^{(l)}}, \tag{4.18}$$

where α is the learning rate (a tunable parameter) and $\partial J(W)/\partial w_{ij}^{(l)}$ is the derivative of the cost function of the neural network with respect to the weight $w_{ij}^{(l)}$. Effectively, at each iteration the gradient descent algorithm updates all the weights by moving them in the direction of the negative of the gradient at that iteration. The learning rate α defines the size of the step in that direction.

The most commonly used variant of gradient descent that is used in practice is called stochastic gradient descent (SGD) [71]. SGD uses a small subset of training data within a sub-iteration of every iteration. 'A strength of SGD [methods] is that they are simple to implement and also fast for problems that have many training examples' [72]. Effectively, in SGD the weight update equation is the same as Equation 4.18, but it is applied to only a few training examples at a time. The gradient $\partial J(W)/\partial w_{ij}^{(l)}$ is computed every time using a few training examples. Each set of these few training examples is called a *mini-batch*. To give a concrete example, if there are 1000 training examples in total and the mini-batch size is 100, then Equation 4.18 will be applied 10 times to 100 training examples at a time, at each single iteration of SGD. In order to avoid bias caused by the ordering of the training data, it is common practice to randomly shuffle the training data at the beginning of each iteration, before splitting the data into mini-batches.

A further modification of SGD includes what is called *momentum*. Momentum works as follows. Each weight is associated with a *velocity*. We will denote the velocity of weight $w_{ij}^{(l)}$ as $v_{ij}^{(l)}$. For each mini-batch there are two update equations; one for updating the velocity and one for updating the weights. The update equations are the following [71]:

$$v_{ij}^{(l)} := \gamma v_{ij}^{(l)} + \alpha \frac{\partial J(W)}{\partial w_{ij}^{(l)}}, \tag{4.19}$$

$$w_{ij}^{(l)} := w_{ij}^{(l)} - v_{ij}^{(l)}. \tag{4.20}$$

The parameter $\gamma \in (0, 1]$ is the momentum. Combining Equations 4.19 and 4.20, we get the following equation, for a more direct comparison with 4.18:

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \gamma v_{ij}^{(l)} - \alpha \frac{\partial J(W)}{\partial w_{ij}^{(l)}}. \tag{4.21}$$

4.8 Partial derivatives in deep neural networks: Backpropagation

To apply the gradient descent algorithm we need to compute the partial derivatives of the cost function $J(W)$ with respect to all the weights in the neural network. This is done using the *backpropagation* algorithm. The key insight for the backpropagation algorithm comes from observing the forward propagation equations (Equations 4.11-4.16). Each layer's output is the next layer's input. Therefore, the chain rule of derivatives can be applied. And since the cost is computed on the last layer, we can apply the chain rule backwards, starting from the output layer. Dividing the operations into dot products and activations, to start, we only need the partial derivative of the cost with respect to the activation of the output layer.

The cost function of the output layer with respect to the activations is always known. The most commonly used cost function is the squared error. The cost associated with the activation of the l-th unit of the output layer (which corresponds to the l-th class) is:

$$\frac{1}{2}\big(1\{y = l\} - a_l^{(out)}\big)^2, \tag{4.22}$$

where $1\{\cdot\}$ is the indicator function. The derivative of the squared error is:

$$\frac{\partial J(W)}{\partial a_l^{(out)}} = 1\{y = l\} - a_l^{(out)}. \tag{4.23}$$

Let us now take the example of the 2-hidden layer neural network from Section 4.6 on forward propagation. Now that we have calculated the derivative of the output layer with

respect to its activations we can work backwards towards the input using the forward propagation equations. Our goal is to derive equations for the partial derivatives of the cost with respect to the weights of each layer.

For our 2-layer neural network the backpropagation equations are as follows (4.6):

$$\frac{\partial J(W)}{\partial z_l^{(out)}} = \frac{\partial J(W)}{\partial a_l^{(out)}} \frac{\partial a_l^{(out)}}{\partial z_l^{(out)}} = \frac{\partial J(W)}{\partial a_l^{(out)}} g'(z_l^{(out)}) \tag{4.24}$$

$$\frac{\partial J(W)}{\partial w_{kl}^{(out)}} = \frac{\partial J(W)}{\partial z_l^{(out)}} \frac{\partial z_l^{(out)}}{\partial w_{kl}^{(out)}} = \frac{\partial J(W)}{\partial z_l^{(out)}} a_k^{(2)} \tag{4.25}$$

$$\frac{\partial J(W)}{\partial a_k^{(2)}} = \sum_{l \in L_{out}} \left(\frac{\partial J(W)}{\partial z_l^{(out)}} \frac{\partial z_l^{(out)}}{\partial a_k^{(2)}} \right) = \sum_{l \in L_{out}} \left(\frac{\partial J(W)}{\partial z_l^{(out)}} w_{kl}^{(out)} \right) \tag{4.26}$$

$$\frac{\partial J(W)}{\partial z_k^{(2)}} = \frac{\partial J(W)}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial z_k^{(2)}} = \frac{\partial J(W)}{\partial a_k^{(2)}} f'(z_k^{(2)}) \tag{4.27}$$

$$\frac{\partial J(W)}{\partial w_{jk}^{(2)}} = \frac{\partial J(W)}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial w_{jk}^{(2)}} = \frac{\partial J(W)}{\partial z_k^{(2)}} a_j^{(1)} \tag{4.28}$$

$$\frac{\partial J(W)}{\partial a_j^{(1)}} = \sum_{k \in L_2} \left(\frac{\partial J(W)}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial a_j^{(1)}} \right) = \sum_{k \in L_2} \left(\frac{\partial J(W)}{\partial z_k^{(2)}} w_{jk}^{(2)} \right) \tag{4.29}$$

$$\frac{\partial J(W)}{\partial z_j^{(1)}} = \frac{\partial J(W)}{\partial a_j^{(1)}} \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}} = \frac{\partial J(W)}{\partial a_j^{(1)}} f'(z_j^{(1)}) \tag{4.30}$$

$$\frac{\partial J(W)}{\partial w_{ij}^{(1)}} = \frac{\partial J(W)}{\partial z_j^{(1)}} \frac{\partial z_j^{(1)}}{\partial w_{ij}^{(1)}} = \frac{\partial J(W)}{\partial z_j^{(1)}} x_i \tag{4.31}$$

The partial derivatives of the cost function with respect to the weights in each layer are given by Equations 4.31 (for the weights from the input to the first hidden layer), 4.28 (for the weights from the first hidden layer to the second hidden layer) and 4.25 (for the weights from the second hidden layer to the output layer). As with the forward propagation equations, the equations that correspond to the second hidden layer (Equations 4.26-4.28) can be generalised to apply to neural networks with more than 2 layers. If a network has H hidden layers there are going to be $H - 1$ equations like the triplet of Equations 4.26-4.28.

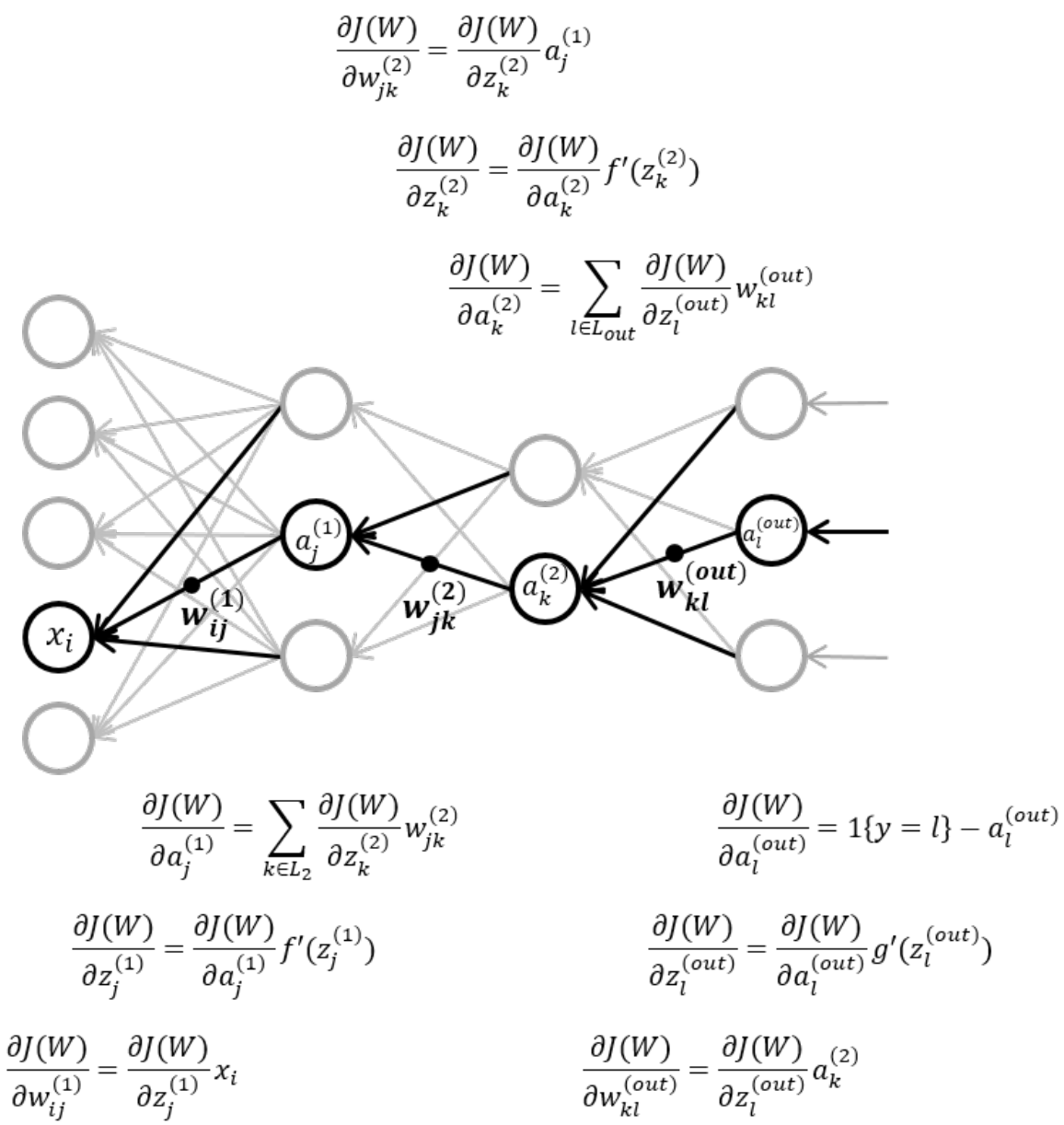

Figure 4.6: Backpropagation

4.9 Regularisation

A common addition to the cost function of deep neural networks is a regularisation or *weight decay* term. The most common type of regularisation is l^2-norm regularisation, which, effectively, is the addition of the sum of the square of all the weights in the network to the cost function. If $w_{ij}^{(l)}$ is the weight from the i-th unit of the $(l-1)$-th layer to the j-th unit of the l-th layer, there are L layers in the network, and there are N_l units in

layer l, the regularisation term which is added to the cost function takes the form:

$$\lambda \sum_{l=1}^{L} \sum_{i=1}^{N_{l-1}} \sum_{j=1}^{N_l} \left(w_{ij}^{(l)} \right)^2. \tag{4.32}$$

The parameter λ is called a regularisation parameter or weight decay parameter, and it controls the relative importance of the regularisation term compared to the cost function. The regularisation term is used for decreasing the magnitude of the weights, which prevents overfitting [71].

4.10 Stacked sparse autoencoders

Stacked sparse autoencoders [7] are a neural network model variant. The key difference between stacked autoencoders and the standard neural networks described in the previous sections is greedy layer-wise pre-training before finetuning the network as a whole. Pre-training followed by finetuning with backpropagation has been shown to result in substantial performance improvements over finetuning using random initialisations for the hyperparameters of the network [9]. In this section we will give an outline of stacked sparse autoencoders (SSAEs). In this section we closely follow the exposition in the excellent tutorial in [71].

Layer-wise pre-training is based on a neural network architecture that is called an *autoencoder*. An autoencoder is an unsupervised learning model for automatically learning features from unlabelled data. An autoencoder is a three-layer neural network (with a single hidden layer) whose goal is to reproduce its input. In a trained autoencoder the activations of the third layer should closely reproduce its input. As in standard neural networks, the activations of the second layer of an autoencoder can be regarded as features automatically learned from the input. These features can in turn be used as input to a second autoencoder that learns second-order features, and so on. It has been shown that this greedy layer-wise training produces good initialisations for the parameters of the

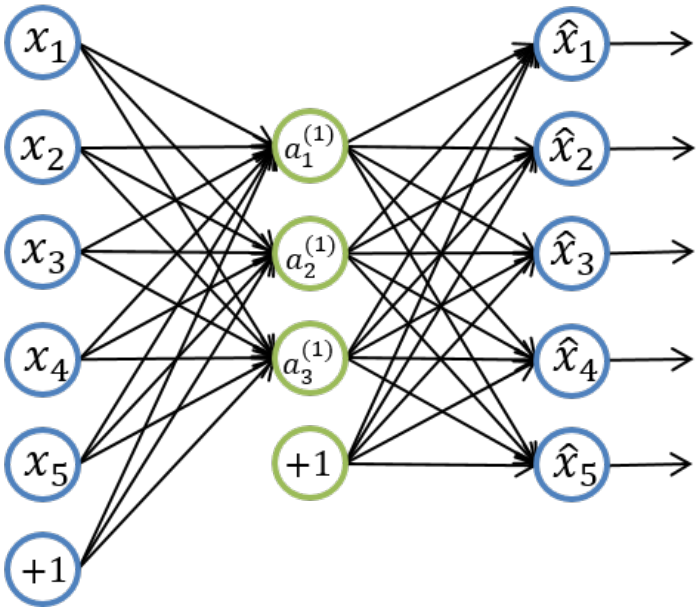

Figure 4.7: Autoencoder for pre-training layer 1 of the network of Figure 4.4.

final neural network that result in better performance than random initialisation [9, 7]. In Figure 4.7 we show an autoencoder with 5 units in the input and output layers and 3 units in the hidden layer.

To train an autoencoder we only need the training data without the labels, i.e. only $\mathbf{x}$ and not y. We will denote as $\hat{\mathbf{x}}$ the activations of the output layer of the autoencoder. As shown in Figure 4.7, the task of the autoencoder is to reproduce its input $\mathbf{x}$ with its output $\hat{\mathbf{x}}$. Since our aim is to reproduce $\mathbf{x}$ by $\hat{\mathbf{x}}$, the basis of the cost function J of the autoencoder is:

$$J(W) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left\| \hat{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)} \right\|_2^2, \tag{4.33}$$

where $\hat{\mathbf{x}}^{(i)}$ are the activations of the third layer for the i-th training example, and $\|\cdot\|_2$ is the l^2 (or Euclidean) norm.

We also apply a regularisation term to the cost function (see Section 4.9). Denoting $w_{ij}^{(l)}$ as the weight from the i-th unit of the $(l-1)$-th layer to the j-th unit of the l-th layer, L the number of layers, and N_l as the number of units in layer l, the cost function

from Equation 4.33 can be extended (see Equation 4.32):

$$J_{reg}(W) = J(W) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{N_{l-1}} \sum_{j=1}^{N_l} \left(w_{ij}^{(l)} \right)^2 \tag{4.34}$$

An autoencoder tries to reproduce its input, by learning an approximation to the identity function. This is a trivial problem unless there are constraints placed on the network. The first constraint that can be applied is restricting the number of nodes s_2 in the second layer to be smaller than the input size. The second restriction that can be applied is a sparsity constraint on the activations of the hidden units. Applying such a sparsity constraint converts an autoencoder into what is called a *sparse autoencoder*.

A way to apply such a sparsity constraint is to demand that the average (over the training set) activation of a unit in the second layer is a small value close to zero, which is the sparsity parameter ρ [71]. We denote the vector of average activations of the i-th unit of the second layer as $\overline{a_i^{(1)}}$. The sparsity constraint is the Kullback-Leibler divergence (KL) between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\overline{a_i^{(1)}}$:

$$\mathrm{KL}(\rho || \overline{a_i^{(1)}}) = \rho \log \frac{\rho}{\overline{a_i^{(1)}}} + (1-\rho) \log \frac{1-\rho}{1-\overline{a_i^{(1)}}}. \tag{4.35}$$

The cost function from Equation 4.34 can then be extended to give:

$$J_{sparse}(W) = J_{reg}(W) + \beta \sum_{i=1}^{N_1} \mathrm{KL}(\rho || \overline{a_i^{(1)}}), \tag{4.36}$$

where β controls the relative importance of the sparsity penalty term.

Training a sparse autoencoder involves the minimisation of $J_{sparse}(W)$ with respect to the parameters W. This is done using the backpropagation algorithm (Section 4.8).

Taking the neural network of Figure 4.4 as an example, we can convert it into a stacked sparse autoencoders model by training an autoencoder for its first hidden layer (Figure 4.7), then using the activations of the first hidden layer as input to an autoencoder for the second autoencoder (Figure 4.8), and, finally, using the activations of the second hidden

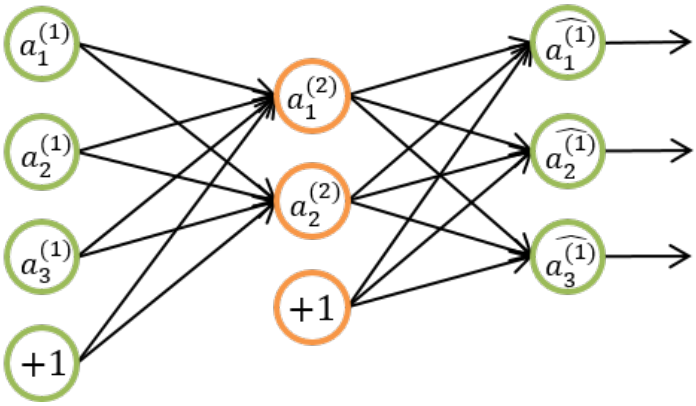

Figure 4.8: Autoencoder for pre-training layer 2 of the network of Figure 4.4.

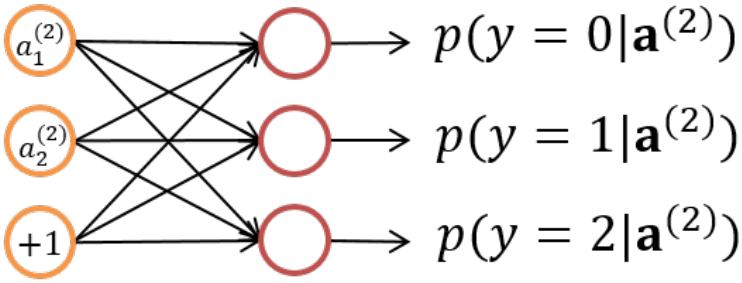

Figure 4.9: Softmax classifier for pre-training the softmax layer of the network of Figure 4.4.

layer as input to a softmax regression classifier (Figure 4.9). After pre-training the two autoencoders and the softmax classifier, the learned parameters can be used to initialise the full neural network (Figure 4.4), and finetune it as a whole.

Chapter 5

Convolutional neural networks

As Goodfellow *et al.* state in their recently published textbook on deep learning, 'convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers' [39, Ch. 9]. In this chapter we introduce convolutional neural networks (CNNs) to make clear how one can go from the neural networks presented in Chapter 4 to CNNs, and what the motivations behind this transitions are. We also aim to show the connection between CNNs and signal decomposition methodologies. This will motivate the analysis and visualisation of CNNs applied to sleep stage scoring in Chapter 7.

We begin with a description of the principles for feature engineering in general, without using CNNs. It is those same principles that form the motivation for CNNs, along with inspiration from information processing in the animal and human visual cortex. Given a classification task, CNNs unify feature engineering and classification into a single algorithm for *end-to-end learning*. We will continue with an overview of the application of CNNs in the field of biomedical engineering which is of particular interest for this thesis.

Finally, following the structure of Chapter 4, we will present the minimal CNN, which is the building block from which, combined with pooling, deep CNN architectures can be constructed. We conclude the chapter with a presentation of deep CNNs, along with the forward propagation and backpropagation equations for CNNs with one-dimensional

input, as this is relevant to biosignal-based applications.

5.1 Principles of feature engineering for classification

The main reason that it is possible to create hand-engineered features is that data like images or biosignals have *local structure*. This way, a feature engineer can design his or her feature extraction methods to leverage this local structure by finding characteristic local patterns.

There are two conditions for the notion of 'locality' to exist in any data:

1. The data must be *homogeneous*. This means that each variable in the data must be of the same type. For example, in an image every variable is a pixel, and in an EEG recording every variable is a measurement of electrical activity.

2. The data must be *continuous*. This means that each variable must have the same distance with its preceding and succeeding variables across one or more dimensions. For example, in an image every pixel is exactly one pixel away from its 8 neighbouring pixels, and in an EEG recording every measurement of electrical activity is exactly $1 \div$ sampling rate seconds away from its two neighbouring measurements.

Finally, these local patterns can only be useful if they are characteristic of multiple localities. This is also called the *location invariance* property in the machine learning literature [59]. In fact, if local patterns always appeared only in specific parts of the data, it could be more effective to use the whole data as is instead of trying to discover local patterns.

Naturally, in homogeneous and continuous data if local structure is to be useful for feature extraction the patterns must be identified in scales smaller than the data. For example, in a 64-by-64 image these local patterns could exist in 8-by-8 or 16-by-16 regions of the full image, and in an EEG recording of duration of 30 seconds local patterns could be found at 1 or 2-second segments of the recording.

For classification problems local patterns are useful if one or more of the following three conditions hold:

1. They appear more frequently in some classes than others;

2. they appear at different locations in different classes;

3. they appear in different combinations in different classes.

The key idea is that local patterns become useful only when they are viewed in conjunction with the classification problem one is trying to solve. No feature can be characterised as useful *a priori*. In a two-stage machine learning workflow, feature extraction is performed independently of classification. Therefore, finding features that work well for the problem at hand is a very time-consuming manual process, which demands going back and forth from feature extraction to classification. The reason is that the performance in a classification task depends on the *combination* of the specific features that have been extracted and the classification algorithm. However, there is no direct feedback between the classification performance and the chosen features apart from the classification metric (e.g. the misclassification rate) itself. CNNs address this particular problem.

5.2 The motivation for CNNs

Convolutional neural networks or CNNs can be intuitively motivated using a specific problem setting in computer vision: image classification. Each image consists of pixels. The 'raw' input to any image classification machine learning pipeline is the greyscale or colour value (usually in the form of three RGB values) of each pixel. For simplicity let us assume greyscale images so that each pixel has a single value. In what follows we will use the term neural networks to refer to standard (non-CNN) neural networks, as the ones presented in Chapter 4.

Because of their capability to learn highly nonlinear hypotheses, deep neural networks are powerful classifiers. Their major drawbacks are the need for large amounts of training

data not only relative to the input size, but also the network size (number of hidden layers and units per layer), and the closely related problem that training is notoriously time-consuming. As we will see, these problems are exacerbated in the application domain of image classification.

We will describe a prototypical application of CNNs, hand-written digit recognition [60]. Consider a very small square image, of size of 64 pixels that contains a hand-written digit. Even this small image contains $64 \times 64 = 4096$ pixels, meaning 4096 input variables for our classification problem. Assume that the digit in each image is not arbitrarily placed but is approximately centred inside the image. Although this makes the problem easier, the variability in the way different people, or even the same person at different times, write these digits is very high. This means that among images of the same digit very few pixels will have the same value. This, in turn, means that the amount of training data that is needed is very large, if we consider the possible combinations of pixels.

Traditionally, computer vision engineers would perform feature extraction before feeding the data into a machine learning classifier. Engineering features has been one of the most active research areas in computer vision in the last decades. By designing those hand-engineered features the engineers were effectively trying to reduce the dimensionality of the input to their machine learning algorithm by extracting features that were relevant to their problem. The major drawback of that two-stage workflow is that there is no rigorous feedback between the two stages. Moreover, engineering new features or tuning existing features by hand for every new application domain is time-consuming.

CNNs achieve a similar effect to the above workflow by automating feature extraction. As we showed in Chapter 4, the activations of the hidden layers of a neural network can be viewed as features. This means that neural networks can learn features that are specific to the classification problem at hand. However, in standard neural networks these features are necessarily global. Effectively, each unit of a hidden layer is fed all of the input from the previous layer. CNNs are the result of using a neural network architecture, while, at the same time, offering the capability to learn local patterns from data.

The conditions for using a CNN are essentially the same as for extracting hand-engineered features: the input data must have local structure with location invariance and be homogeneous and continuous. If, for example, the input variables are different measurements from patient records, the data is heterogeneous. In this case, we cannot use a CNN. If, on the other hand, the input is a natural image, it is both homogeneous and continuous, and exhibits local patterns with location invariance. Therefore, going back to the image classification problem, CNNs can be used to reduce the number of parameters in a neural network, by exploiting the local structure of images. Then, we can benefit from the capability of neural networks to learn highly nonlinear hypotheses, and end-to-end learning, while reducing the training time to more acceptable levels. Concretely for 64-by-64 images, a regular neural network would have 4096 $\times$ the number of units in the hidden layer parameters for the first hidden layer. A CNN with 8-by-8 filters in the first hidden layer would only have 64 $\times$ the number of units in the hidden layer parameters for the first hidden layer. Later in this chapter we will show exactly how this is achieved.

As LeCun *et al.* note [59], CNNs were inspired by the hierarchical structure of information processing in the visual cortex [46, 47, 31]. Specifically, a physical view of convolution can be given by examining the receptive fields of the neurons in the visual cortex. In 1959, Hubel and Wiesel [46] published their pioneering work on the receptive fields of the neurons of the cat's primary visual cortex. Receptive fields can be seen as 'filters' that detect light of particular shape, as defined by the excitatory or 'on' areas. The more similar the light's shape is to the excitatory areas and the less it overlaps with the inhibitory areas, the higher the frequency of firing in the corresponding neuron is. The incoming light can be viewed as the signal and the receptive field as the filter. The frequency of firing can be seen as the dot product between the incoming light and the filter.

5.3 CNNs in biomedical engineering

In recent years, the machine learning community has witnessed an explosion in the use of convolutional neural networks (CNNs). Researchers have achieved unprecedented success using CNNs, most notably in computer vision in areas such as object recognition (e.g. [56]), image segmentation (e.g. [36]) and face recognition (e.g. [92]). The key to the success of CNNs has been end-to-end training, i.e. the integration of feature extraction and machine learning into a single algorithm using the 'raw' data as input (pixels, in the case of computer vision).

CNNs are trained to learn features specific to the machine learning task at hand. This is achieved by learning features which must necessarily improve the performance in the metric of interest (e.g. the misclassification rate) in a single step. In other words, no feature is learned (or used) unless it improves the performance in the learning task. This is the main advantage of CNNs over the common two-step pipeline of signal processing for feature extraction and machine learning using the features from the first step.

The content of the images used in computer vision research has mainly comprised natural scenes and photos of objects or people. The reason is that this type of images are relevant to applications with large commercial applicability, such as web search, self-driving cars, social networking and security systems. Nevertheless, the advances in CNNs for computer vision have been rapidly transferred into the biomedical field in applications that are also based on two-dimensional images (including sequences of images in time), most notably medical imaging (e.g. [19]). CNNs proved to be a very powerful generic algorithm that can be adapted for any type of image with relatively few modifications in the overall architecture of the neural network.

Image-based applications constitute a large part of biomedical applications, particularly with the widespread adoption of modern imaging technologies such as functional magnetic resonance imaging (fMRI). Nevertheless, an equally large part of biomedical applications is based on one-dimensional biosignals. The most common biosignals are

the electroencephalogram (EEG), the magnetoencephalogram (MEG), the electrocardiogram (ECG), the electromyogram (EMG), the electrooculogram (EOG), the galvanic skin response (GSR), the respiration rate, and actigraphy using accelerometers.

The analysis of biosignals has applications in the diagnosis and monitoring of a diverse number of diseases, such as epilepsy, sleep disorders, heart problems, psychiatric disorders, and neurodegenerative diseases. Applying state-of-the-art machine learning techniques to these areas has tremendous potential for automating the diagnosis and monitoring of many diseases that are very expensive and inefficient to diagnose and monitor using existing technology.

CNNs are one of the most promising state-of-the-art machine learning algorithms for biomedical engineering. Furthermore, CNNs are directly applicable to one-dimensional biosignals and produce interpretable results. However, the overwhelming majority of programming libraries, reports and papers take an almost exclusively image-based perspective. This has created the false impression that CNNs are suitable only for image-based machine learning applications, and slowed down their adoption by the entire biomedical engineering community. The slow adoption of CNNs for one-dimensional signals is all the more surprising given that the first industrial application of CNNs was not in computer vision but in automatic speech recognition [42] (speech signals are also one-dimensional).

Another reason for the slow adoption of CNNs for biosignal-based applications is that images are by their nature more intuitively perceived and understood. When a CNN learns a filter for an image-based application, the filter is also an image, which can be visualised and provide intuition about the effect it has, in its raw, pixel-level form without further analysis. In one-dimensional signals, on the other hand, this is not the case, and signal processing techniques must be used to interpret the learned filters. However, the machine learning and signal processing communities do not necessarily overlap in terms of the methods that they employ.

Despite the above, there has recently been a small but growing interest in using CNNs for biosignal-related problems [82, 17, 55, 103], including a 3rd place in a very competitive

machine learning competition on the Kaggle platform [53] with EEG signals.

5.4 The minimal CNN

In general, a CNN is a neural network that can make use of the local structure in homogeneous and continuous data. In this section, we will describe the minimal CNN which is an extension to the minimal neural network. As LeCun, Bengio and Hinton say [59] 'there are four key ideas behind CNNs that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers'. In this section, we will introduce the first two key ideas.

In Figure 5.1 we show the minimal CNN. It is a neural network with a single hidden layer, which contains a single unit. As we will show, the units in the hidden layer perform convolution (it is denoted by the $*$ symbol). This is why hidden layers of this type are called convolutional. The difference between a unit of a hidden layer of a regular neural network and a unit of a convolutional layer is that the unit of the convolutional layer outputs more than one activation. This is the reason that in Figure 5.1 we chose to represent the activations of the convolutional layer as a separate layer. This representation will prove helpful when we extend our discussion to larger CNNs. The output layer is a logistic regression layer.

In the example of Figure 5.1, there are 5 elements in the input vector, plus a bias element. In Figure 5.1 we break down the computation that happens in the minimal CNN into steps from left to right, to illustrate local connections and weight sharing, the first two key ideas behind CNNs according to [59].

The connections between the input units and the unit of the convolutional layer are local, in the sense that at each step of the computation there are connections from only a subset of the input units to the convolutional unit. Importantly, the weights from the input units to the convolutional unit are shared. Although there are 5 units of input (excluding the bias unit) there are only 3 weights that are shared across the three steps

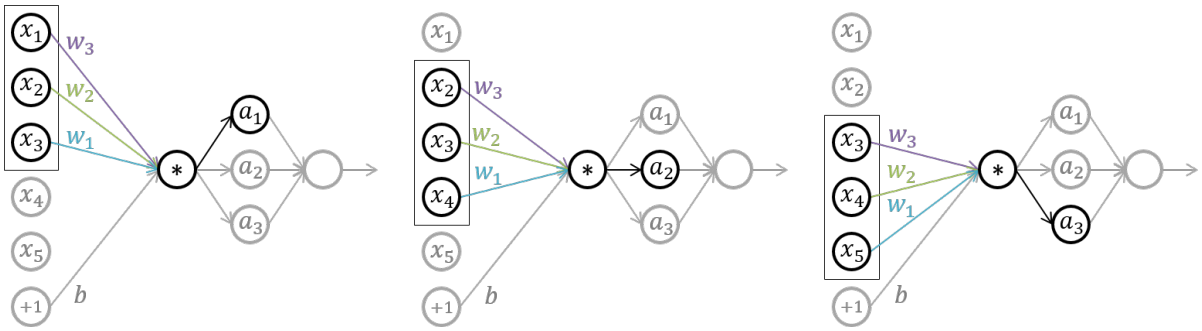

Figure 5.1: The minimal CNN. We show the three parts of the computation that occurs in the minimal CNN from left to right. The first layer is the input, the second layer is a convolutional layer, the third layer is the result of the convolution, and the output layer is logistic regression.

of the computation. The first weight always corresponds to the first unit in the local triplet of units, the second weight to the second unit, and the third weight to the third unit.

Let us break down the three parts of the computation into three equations (omitting the bias unit for clarity):

$$a_1 = f(w_3 x_1 + w_2 x_2 + w_1 x_3) \tag{5.1}$$

$$a_2 = f(w_3 x_2 + w_2 x_3 + w_1 x_4) \tag{5.2}$$

$$a_3 = f(w_3 x_3 + w_2 x_4 + w_1 x_5) \tag{5.3}$$

where a_t is the activation of the output unit for the local input starting at time t.

The entire computation in Equations 5.1-5.3 is essentially a sliding dot product. As we saw in Chapter 3, a sliding dot product (with a flipped filter) is a convolution. The definition of convolution (Equation 3.6) requires that the kernel be flipped. Let us define the vector of weights as flipped, i.e. $\mathbf{w} = (w_3, w_2, w_1)$, so that the last weight of the neural network is the first element of $\mathbf{w}$, and so on. Following the notation of Chapter 3, the number M of elements in vector $\mathbf{x}$ (the signal) is equal to 5, and the number N of elements in vector $\mathbf{w}$ (the filter) is equal to 3. Therefore, the size of the valid convolution is $5 - 3 + 1 = 3$. We can then write Equations 5.1-5.3 using the definition of convolution

in Equation 3.6 as:

$$a_t = f\big((\mathbf{x} * \mathbf{w})_t\big) = f\bigg(\sum_{i=0}^{2} x_{t+1} w_{3-i}\bigg) \text{ for } 1 \leq t \leq 3. \tag{5.4}$$

The activations of the convolutional layer then give us new input, which we can denote as a vector $\mathbf{a} = (a_1, a_2, a_3)$, whose order is meaningful in the same way that the order in the elements of the input was meaningful. Therefore, if input vector $\mathbf{x}$ is a one-dimensional signal, then the activations of the convolutional layer also form a one-dimensional signal. This is an important observation for constructing deep CNNs, which we will describe later. In the minimal CNN, the activations of the convolutional layer are then used as the input of the output layer for classification.

From a signal processing perspective, the vector $\mathbf{w}$ of weights from the input to the convolutional layer is a filter. Similarly to the vector $\mathbf{a}$ of activations, the order of the elements of the weight vector $\mathbf{w}$ is meaningful.

Finally, we should note that the length of the filters depends on the specific application.

5.5 Pooling

The third key characteristic of CNNs according to [59] is pooling. The motivation for pooling is the following. In the example of the minimal CNN in Figure 5.1 the input layer had size $M = 5$ and the number of local connections to the convolutional layer (or, equivalently, the size of the filter) was $N = 3$. The number of activations (or, equivalently, the size of the result of the convolution between the filter and the input) was $M - N + 1 = 3$. These numbers were used for illustration reasons only, as they are not representative of real-world applications. In real world applications it is common for the input to be a 10 second long signal sampled, for instance, at 256 Hz. In this case the size of the input would be $M = 2560$. Let us assume a filter with length of 1 second, so that $N = 256$. In this case the size of the result of the convolution would

be $M - N + 1 = 2305$. In this case the output of the convolutional layer is almost as large as the input. Moreover, in real-world CNNs there are a lot more than one units in a convolutional layer, so that, if there were 50 units in the convolutional layer the total number of activations would be $2305 \times 50 = 115250$. Clearly, using a convolutional layer explodes the size of the feature space, so that the problem of having a very large input is worsened. The considerations with that are not only computational. Using the entire output of the convolutional layers would make the neural network learn extremely fine-grained variations in the filtering of the input. Ideally, we would like the network to exhibit some *translation invariance* to small local shifts of the input in time or space. As Goodfellow *et al.* state, '[i]nvariance to local translation can be a very useful property if we care more about whether some feature is present than exactly where it is' [39, Ch. 9, p. 342]. Similarly, as LeCun, Bengio and Hinton state, 'the role of the pooling layer is to merge semantically similar features into one' [59]. Location invariance can therefore both prevent overfitting, and reduce the amount of data and computation time that is needed to train a CNN.

Pooling is a technique that addresses the above considerations. Pooling works by combining ('pooling') nearby activations (outputs of a convolutional layer) by substituting multiple activations in a neighbourhood with a summary statistic. Pooling can be also interpreted as a simple denoising operation. In CNN practice the two most commonly used pooling procedures are mean pooling and max pooling. In mean pooling we take the mean of nearby activations, and in max pooling the maximum. We can see that for small translations of the output those pooling functions are expected to produce similar or potentially identical summary statistics. There are two parameters that need to be set for pooling: the pooling region size and the pooling step. The pooling region size is the number of activations that will be pooled, and the pooling step defines whether there will be overlapping pooling regions or not. The pooling region size must be larger than one, and up to the length of the activations vector. For the pooling step we have: 1 $\leq$ pooling step $\leq$ pooling region size. When the pooling step is equal to the size of the

pooling region, there are no overlapping regions. When the pooling step is equal to 1, we have maximum overlap.

Continuing with our previous example, if we choose a pooling region of size 5 and a pooling step of size 5, we will have 461 non-overlapping pooling regions (this is the size of the result of convolution divided by the size of the pooling region, i.e. $2305 \div 5 = 461$). This reduces the size of the activations vector by 5 times. In theory, it is possible to increase the pooling region size to a very large number (even as large as the length of the activations vector), however, this is not always desirable, as useful information for the problem at hand might be lost. Effectively, if the pooling region size is very large, we pool over the whole result of the convolution, and, therefore, we lose time-specific information, e.g. at which points the activation which corresponds to a particular filter was large. For many applications this information is useful.

5.6 Deep CNNs

Deep CNNs are CNNs with multiple convolutional layers. The key observation for constructing deep CNNs is that the output of a single convolutional layer (either before or after pooling) is itself a signal, which is continuous and homogeneous, as its input is (if the input did not have these characteristics we would not have been able to use a convolutional layer in the first place). Therefore, we can add a second convolutional layer that takes the activations of the first convolutional layer as its input.

An important observation is that the first convolutional layer outputs as many activation vectors as its hidden units. So if there are 3 units in a convolutional layer its output will be 3 activation vectors. In that case, the input to the second convolutional layer will be 3 different signals. If the second convolutional layer has 2 units, we will have 6 activation vectors as the output, one for each signal-unit combination. This is important, because as we add convolutional layers with pooling the size of each activation vector gets smaller, but the number of activation vectors gets larger (unless there is only

a single unit in the convolutional layer, in which case it stays the same; but this is a very unlikely case).

In a deep neural network, there is usually one or more "regular" hidden layers between the last convolutional layer and the output layer. Regular hidden layers are also called fully connected layers in the literature. Deep learning pioneer Yann LeCun views fully connected layers as just a special case of convolutional layers [58]. Effectively, fully connected layers are just convolutional layers in which the input has the same size as the filters, or, as LeCun puts it, each unit does a 1×1 convolution.

We will now continue with an example of a deep CNN. We will omit the bias units to keep the figure and exposition uncluttered. In Figure 5.2 we show this deep CNN. It has an input layer, two convolutional layers, each followed by a pooling layer, one fully connected hidden layer, and an output layer. In the figure, we highlighted a single instance of the operation that each layer performs. In particular, for a convolutional layer we highlight a single dot product of a single unit.

The input layer of the CNN has 15 units, i.e. the input is a signal with 15 timepoints. The first convolutional layer, C1, has 3 units, i.e. there are 3 different filters in it. The size of each filter is 4 timepoints, indicated with a rectangle on the input layer. We show only the weights from the first 4 units of the input to each unit in C1, i.e. the first step of the convolution operation. Each unit in C1 outputs a new signal (the result of convolution) with $15 - 4 + 1 = 12$ timepoints, shown as 12 units in the figure.

Each output signal of the C1 is fed into the first pooling layer, P1, with pooling region size of 2. To keep the figure uncluttered we set the pooling step to also be 2, so that there are no overlapping pooling regions. After pooling these signals of length 12 are reduced to a length of 6. There are 3 of these pooled signals, as there are 3 filters in the C1. The output signals of a convolutional layer are also called *feature maps* in the neural network literature.

It is worth pausing here to consider the differences in the number of parameters and connections between a convolutional layer and a fully connected layer. The number of

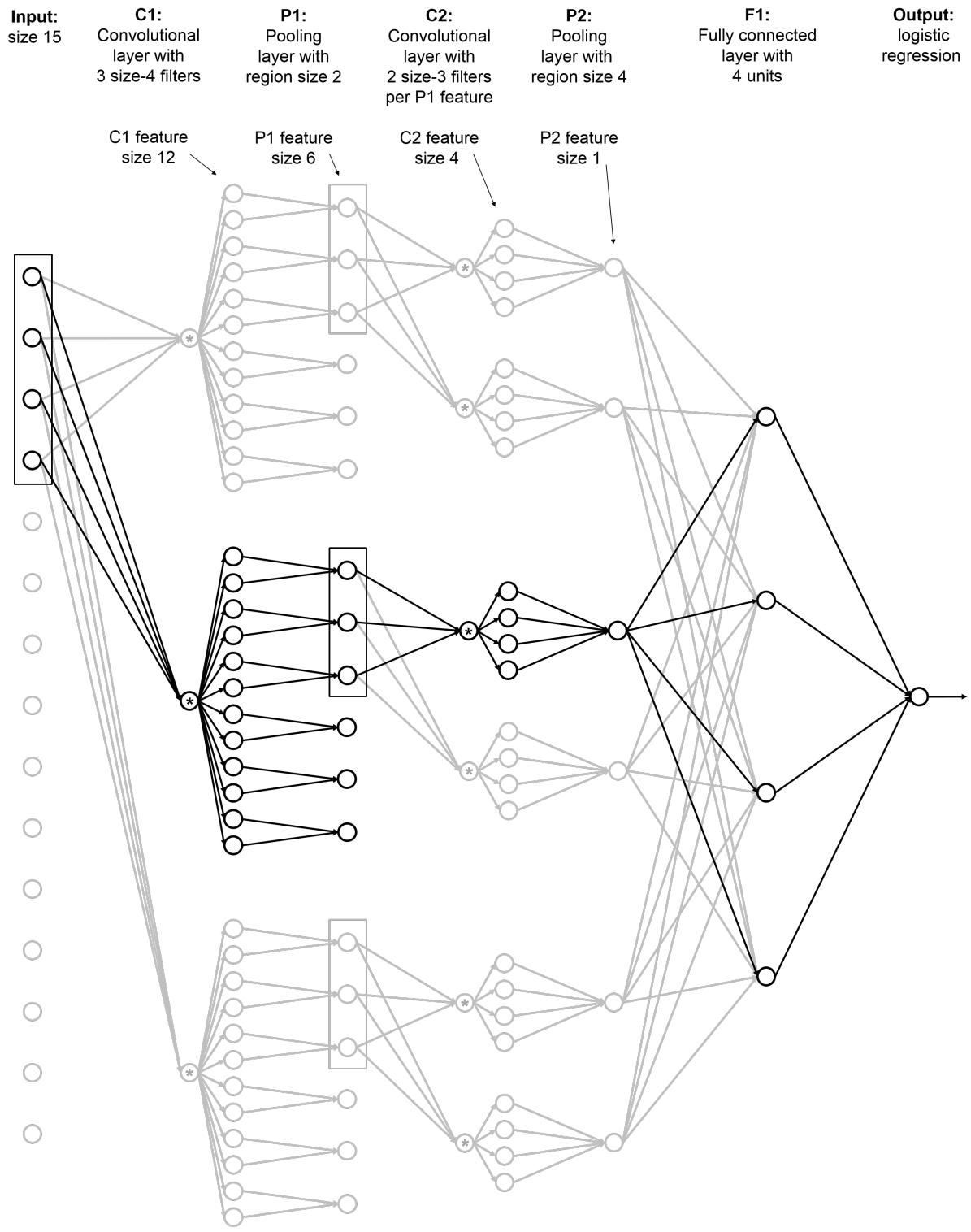

Figure 5.2: A deep CNN with 7 layers: an input layer, two convolutional layers (C1 and C2), two pooling layers (P1 and P2), one fully connected layer (F1) and an output layer. The elements in bold in the figure highlight a single instance of the operation that each layer performs. Convolutional units are denoted with the $*$ symbol.

parameters in a convolutional layer is calculated as: number of inputs $\times$ (filter size + 1) $\times$ number of filters per input, if we add 1 for the bias element. For layer C1 in Figure 5.2 we have a single input (a single signal, not a single unit) and 3 filters of size of 4. This means there are number of inputs $\times$ (filter size + 1) $\times$ number of filters $= 1 \times (4+1) \times 3 = 15$ unique parameters. Remember that the weights to a single convolutional unit are shared between the units of the input, therefore there are only as many weights per filter as the size of the filter, and not the size of the input signal as would be the case for a fully connected layer.

The number of connections in a convolutional layer is given by: number of inputs $\times$ ((size of convolution result $\times$ filter size) + 1) $\times$ number of filters per input. This means that there are $1 \times ((12 \times 4)+1) \times 3 = 147$ connections in layer C1. If layer C1 were a fully connected layer we would have $(15+1) \times 3 = 45$ unique parameters, and the same number of connections. In general, the number of parameters is smaller in a convolutional layer as opposed to a fully connected layer, but the total number of connections is larger in a convolutional layer.

Let us now continue with the second convolutional layer, C2, in the CNN of Figure 5.2. The pooled signals from the first pooling layer, P1, become the input to C2. In our example, C2 has 2 units (filters) per P1 feature. Commonly in the literature the 'per P1 feature' part is omitted, which can create confusion. This is the reason we state it explicitly. This is the most common way for constructing convolutional layers beyond the first, as usually, but not necessarily, the first convolutional layer's input is a single signal.

It is important to note that, in our example, each feature from layer P1 is *independently* fed into each unit of layer C2. There are examples of CNN architectures in which features from a previous convolutional/pooling layer are combined in the next layer (e.g. [60]).

The filters in C2 have a size of 3. Given that the input from P1 has a size of 6, the output of each filter in C2 is a signal of length 4. The second pooling layer, P2, has a pooling region of 4, exactly as the length of its input, so that the output is of each P2 unit has length 1. After layer P2, we have a fully connected layer, F1 with 4 units, and,

finally, an output layer.

5.7 Forward propagation and backpropagation in CNNs

In the previous sections we described how forward propagation works in CNNs. Let us now give all the equations of forward propagation. Consider the CNN in Figure 5.3.

The main difference between a fully connected neural network and a CNN is in the weight sharing of the convolutional layers. To deal with weight sharing with clarity, in our exposition we follow the following conventions. The index i of the input is a *relative*, and not an absolute one. It is relative to the start of the interval that is considered each time as the filter slides on the input. In this sense, it is the index of the filter's element, since the filter remains the same as it slides through the input. The absolute index of the input is given by combining the number j of times the filter has slid through the input with the index i. The index j is the index of the element of the output of the convolutional layer. As we show in Figure 5.3 the absolute index of the input is given by: $(j-1)+i$.

To keep the notation uncluttered, we omitted the index of the convolutional unit in layer C1. We should note, though, that any convolutional operation in the equations that follow refers to a single unit (filter) in the convolutional layer. Assuming a max pooling function for the pooling layer P1, the forward propagation equations are:

$$z_j^{(C1)} = \sum_{i=0}^{|C1|-1} w_{|C1|-i}^{(C1)} x_{j+i} \tag{5.5}$$

$$a_j^{(C1)} = f(z_j^{(C1)}) \tag{5.6}$$

$$a_k^{(P1)} = \max(a_j^{(C1)} \text{ for } j \in \text{pooling region}) \tag{5.7}$$

$$z_l^{(F1)} = \sum_{k \in P1} w_{kl}^{(F1)} a_k^{(P1)} \tag{5.8}$$

$$a_l^{(F1)} = f(z_l^{(F1)}) \tag{5.9}$$

$$z_m^{(out)} = \sum_{l \in F1} w_{lm}^{(out)} a_l^{(F1)} \tag{5.10}$$

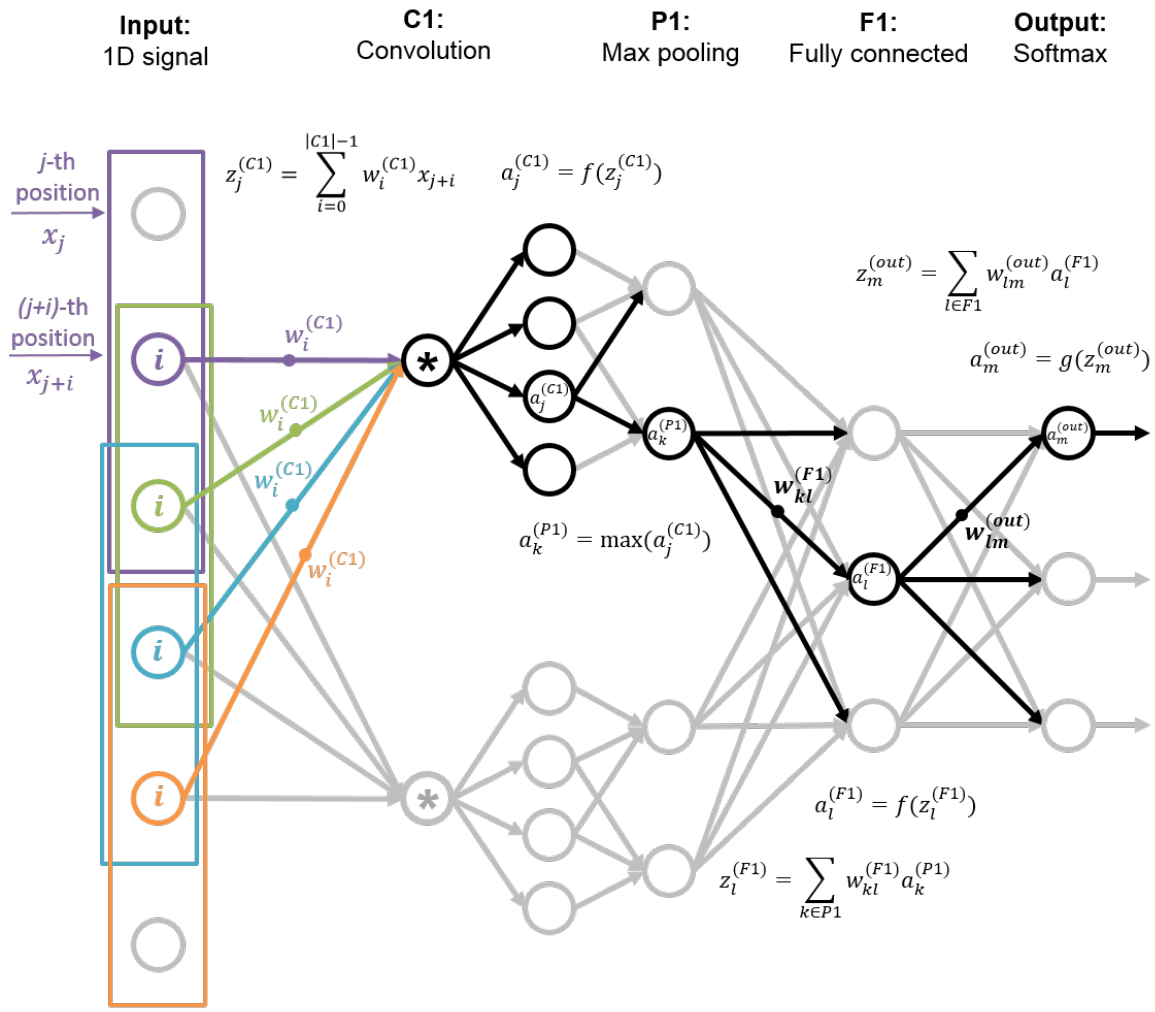

Figure 5.3: CNN forward propagation

$$a_m^{(out)} = g(z_m^{(out)}) \tag{5.11}$$

Equations 5.5 and 5.6 correspond to the convolutional layer C1. Notice that contrary to the case of a fully connected layer (see Equation 4.11 earlier), the sum over i in Equation 5.5 is not over the units of the input, but over the elements of the filter. Notice also, that, as we noted earlier, the indexing of the input is therefore given as a function of the unit of the convolutional layer output. This naturally leads to convolution. Effectively, if we apply Equation 5.5 starting at $j = 1$ and ending at $j = (|in| - |C1| + 1)$ we have performed a valid convolution. This formulation will prove illustrative in our discussion

of backpropagation.

Equation 5.7 corresponds to the pooling layer P1. We denoted the result of the pooling operation as activations. The pooling function can, of course, take any other form, but the most common functions are max pooling (as in Equation 5.7) and mean pooling.

Finally, as in the fully connected case, Equations 5.8 and 5.9 correspond to the fully connected layer F1, and Equations 5.10 and 5.11 correspond to the output layer.

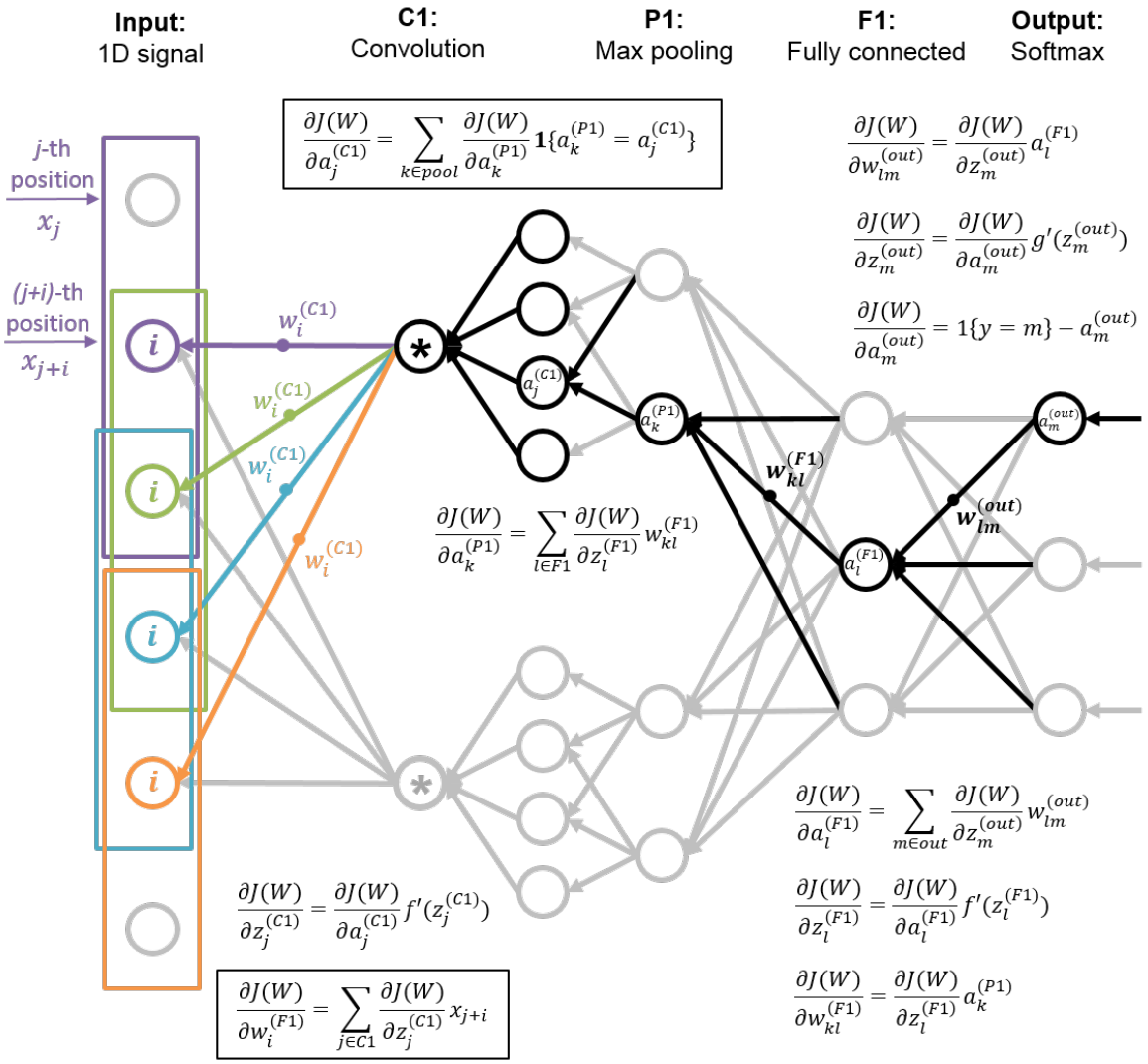

Figure 5.4: CNN backpropagation

We now turn to backpropagation, as shown in Figure 5.4. The first steps of the

backpropagation will be the same as in the fully connected neural network. The two differences will be in the pooling and the convolutional layers.

The backpropagation equations are:

$$\frac{\partial J(W)}{\partial a_m^{(out)}} = 1\{y = m\} - a_m^{(out)} \tag{5.12}$$

$$\frac{\partial J(W)}{\partial z_m^{(out)}} = \frac{\partial J(W)}{\partial a_m^{(out)}} \frac{\partial a_m^{(out)}}{\partial z_m^{(out)}} = \frac{\partial J(W)}{\partial a_m^{(out)}} g'(z_m^{(out)}) \tag{5.13}$$

$$\frac{\partial J(W)}{\partial w_{lm}^{(out)}} = \frac{\partial J(W)}{\partial z_m^{(out)}} \frac{\partial z_m^{(out)}}{\partial w_{lm}^{(out)}} = \frac{\partial J(W)}{\partial z_m^{(out)}} a_l^{(F1)} \tag{5.14}$$

$$\frac{\partial J(W)}{\partial a_l^{(F1)}} = \sum_{m \in out} \left(\frac{\partial J(W)}{\partial z_m^{(out)}} \frac{\partial z_m^{(out)}}{\partial a_l^{(F1)}} \right) = \sum_{m \in out} \left(\frac{\partial J(W)}{\partial z_m^{(out)}} w_{lm}^{(out)} \right) \tag{5.15}$$

$$\frac{\partial J(W)}{\partial z_l^{(F1)}} = \frac{\partial J(W)}{\partial a_l^{(F1)}} \frac{\partial a_l^{(F1)}}{\partial z_l^{(F1)}} = \frac{\partial J(W)}{\partial a_l^{(F1)}} f'(z_l^{(F1)}) \tag{5.16}$$

$$\frac{\partial J(W)}{\partial w_{kl}^{(2)}} = \frac{\partial J(W)}{\partial z_l^{(F1)}} \frac{\partial z_l^{(F1)}}{\partial w_{kl}^{(F1)}} = \frac{\partial J(W)}{\partial z_l^{(F1)}} a_k^{(P1)} \tag{5.17}$$

$$\frac{\partial J(W)}{\partial a_k^{(P1)}} = \sum_{l \in F1} \left(\frac{\partial J(W)}{\partial z_l^{(F1)}} \frac{\partial z_l^{(F1)}}{\partial a_k^{(P1)}} \right) = \sum_{l \in F1} \left(\frac{\partial J(W)}{\partial z_l^{(F1)}} w_{kl}^{(F1)} \right) \tag{5.18}$$

$$\frac{\partial J(W)}{\partial a_j^{(C1)}} = \sum_{k \in pool} \left(\frac{\partial J(W)}{\partial a_k^{(P1)}} \frac{\partial a_k^{(P1)}}{\partial a_j^{(C1)}} \right) = \sum_{k \in pool} \left(\frac{\partial J(W)}{\partial a_k^{(P1)}} \mathbf{1}\{a_k^{(P1)} = a_j^{(C1)}\} \right) \tag{5.19}$$

$$\frac{\partial J(W)}{\partial z_j^{(C1)}} = \frac{\partial J(W)}{\partial a_j^{(C1)}} \frac{\partial a_j^{(C1)}}{\partial z_j^{(C1)}} = \frac{\partial J(W)}{\partial a_j^{(C1)}} f'(z_j^{(C1)}) \tag{5.20}$$

$$\frac{\partial J(W)}{\partial w_{|C1|-i}^{(C1)}} = \sum_{j \in |C1|} \left(\frac{\partial J(W)}{\partial z_j^{(C1)}} \frac{\partial z_j^{(C1)}}{\partial w_{|C1|-i}^{(C1)}} \right) = \sum_{j \in |C1|} \left(\frac{\partial J(W)}{\partial z_j^{(C1)}} x_{j+i} \right) \tag{5.21}$$

The equations that are different from the fully connected case are Equation 5.19, which includes the derivative of the pooling function, and Equation 5.21, which includes the derivative with respect to the shared weights of the convolutional unit. In this example we used the max pooling function (see Equation 5.7), whose derivative is the indicator function in Equation 5.19 (i.e. it is equal to one only for the argument that is the maximum). Equation 5.21 is the derivative of the cost function with respect to the *shared* weights. We used the indexing from forward propagation in Equation 5.5 where the filter is flipped.

Chapter 6

Automated sleep scoring using time-frequency analysis and stacked sparse autoencoders

In this chapter we present the machine learning methodology that we developed for automatic sleep stage scoring using a single channel of EEG [94]. Our time-frequency analysis-based feature extraction is fine-tuned to capture sleep stage-specific signal features as described in the American Academy of Sleep Medicine (AASM) manual that the human experts follow. We used ensemble learning with an ensemble of stacked sparse autoencoders for classifying the sleep stages. We used class-balanced random sampling across sleep stages for each model in the ensemble to avoid skewed performance in favour of the most represented sleep stages, and addressed the problem of misclassification errors due to class imbalance while significantly improving worst-stage classification. Our method has both high overall accuracy (78%, range 75–80%), and high mean F_1-score (84%, range 82–86%) and mean accuracy across individual sleep stages (86%, range 84–88%) over all subjects. The performance of our method appears to be uncorrelated with the sleep efficiency and percentage of transitional epochs in each recording.

6.1 Methodology

There are two main parts in our methodology. We will first describe the feature extraction methods that we used, and then we will give the details of the machine learning methods.

6.1.1 Feature extraction methodology

For feature extraction we performed time-frequency analysis using complex Morlet wavelets, which we described in detail in Chapter 3. Figure 6.1 shows a plot of signals from the Physionet dataset (see Section 2.8). In the figure, one 30-second signal at 100 Hz (3,000 timepoints) per sleep stage is shown.

For time-frequency analysis using complex Morlet wavelets there are two sets of parameters that need to be chosen, the peak frequencies and the number of wavelet cycles per frequency. The number of wavelet cycles defines its width and controls the trade-off between temporal and frequency precision. Specifically, increasing the number of cycles increases the frequency precision but decreases the temporal precision, while decreasing the number of cycles increases the temporal precision but decreases the frequency precision.

In our study we selected the peak frequencies and the number of cycles based on the sleep scoring criteria in Table 2.2, taking into account the transition rules in Table 2.3. The wavelets that we used attempt to capture seven main types of activity: slow waves, K-complexes, mixed delta/theta activity, alpha activity, sleep spindles, beta activity, and gamma activity. To capture slow waves, which are characteristic of stage N3, we used three wavelets with peak frequencies, at 0.7, 1, and 1.5 Hz. To capture K-complexes for stage N2, we used two wavelets with peak frequencies at 2 and 3.2 Hz. Mixed delta/theta activity is prevalent in multiple stages (N1, R, and W), and we used four wavelets at 3, 4, 5, and 6 Hz to capture it. Alpha activity is one of the defining characteristics of stages N1 and W, and we used three wavelets at 8, 10, and 12 Hz to capture it. Sleep spindles occur mostly in stage N2, but also in some stage N3 epochs, and we used wavelets at 12,

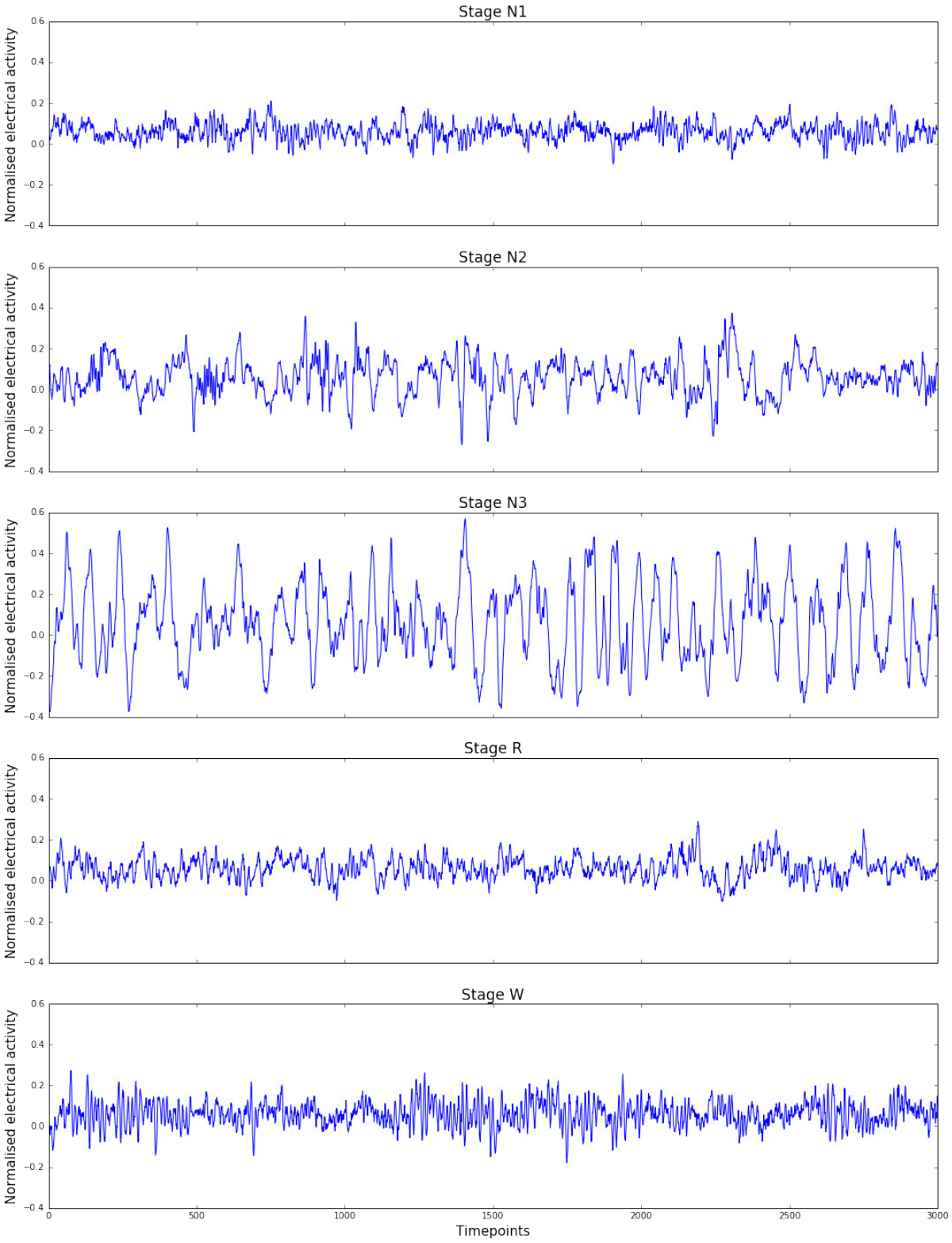

Figure 6.1: A plot of signals from the Physionet dataset. There is one 30-second signal at 100 Hz (3,000 timepoints) per sleep stage.

13, 14, and 15 Hz to capture them. Beta activity is associated with arousals, as a scoring criterion based on stage transitions for N1 (see first and third row in Table 2.3), but also stage W. We capture beta activity with wavelets at 16, 18, and 20 Hz. Finally, we capture gamma activity with a single wavelet at 40 Hz. Gamma activity is not included in the sleep scoring manual, but there is evidence in the literature that features from modalities other than EEG, such as eye movements [101], stage R sleep [52] and EMG activity [38, 97], can manifest themselves in the gamma activity of EEG. Regarding the width of the wavelets, when we were looking for patterns of activity localised in time, such as sleep spindles, we used a small number of cycles (3) for the wavelets, as this localises the detected activity in time, for a frequency precision trade-off. Conversely, when we were looking for sustained activity across the epoch, most notably alpha activity, we used a high number of cycles (10) for the wavelets. In the cases for which no guidelines existed or the guidelines were not specific, such as gamma activity, we used 5 cycles for the wavelets. In Table 6.1 we summarise the parameters chosen.

After extracting the frequency-band power for each peak frequency given in Table 6.1, the features that we computed for each epoch were the power of the frequency-band power signal, the power of the time-domain signal, the Pearson correlation coefficient between each pair of frequency-band power signals and the autocorrelation in the time-domain signal for 50 time lags (i.e. up to 0.5 seconds). Intuitively, the power of the frequency-band power signal measures the strength of the activity at a specific frequency band over the epoch. The power of the time-domain signal is a measure of the oscillation of the signal and the overall amplitude that the signal exhibits across the epoch. The Pearson correlation coefficient between frequency-band power signals aims at capturing correlation between activity at different frequency bands over the epoch. This is useful for epochs in which multiple sleep scoring features appear at the same time. The autocorrelation of the time-domain signal aims at capturing dependencies in the values of the EEG signal at one point in time with previous values. Additionally, we used a sliding window to extract the power of the frequency-band power and the power of the time-domain signal

Table 6.1: Peak frequencies and number of wavelet cycles per frequency for time-frequency analysis using complex Morlet wavelets.

Target Frequency Band	**Target Sleep Stages**	**Frequency or Time Precision**	**Peak Frequency (Hz)**	**Number of Wavelet Cycles**
slow (0.5-2 Hz)	N3	Time	0.7	3
slow (0.5-2 Hz)	N3	Time	1	3
slow (0.5-2 Hz)	N3	Time	1.5	3
K-complex (1.6-4 Hz) [40]	N2	Time	2	3
K-complex (1.6-4 Hz) [40]	N2	Time	3.2	3
delta/theta (2-7 Hz)	N1,R,W	Intermediate	3	5
delta/theta (2-7 Hz)	N1,R,W	Intermediate	4	5
delta/theta (2-7 Hz)	N1,R,W	Intermediate	5	5
delta/theta (2-7 Hz)	N1,R,W	Intermediate	6	5
alpha (8-13 Hz)	N1,W	Frequency	8	10
alpha (8-13 Hz)	N1,W	Frequency	10	10
alpha (8-13 Hz)	N1,W	Frequency	12	10
spindle (12-15 Hz)	N2,N3	Time	12	3
spindle (12-15 Hz)	N2,N3	Time	13	3
spindle (12-15 Hz)	N2,N3	Time	14	3
spindle (12-15 Hz)	N2,N3	Time	15	3
beta (15-30 Hz)	N1 (arousal)	Time	16	3
beta (15-30 Hz)	N1 (arousal)	Time	18	3
beta (15-30 Hz)	W	Intermediate	20	5
gamma (30-100 Hz) *	N1,N2,N3,R,W	Intermediate	40	5

* There is evidence in the literature that features from modalities other than EEG, such as eye movements [101], stage R sleep [52] and EMG activity [38, 97], can manifest themselves in the gamma activity of EEG.

at different intervals within each epoch. Specifically, we used a sliding window of duration of 5 seconds and step of 2.5 seconds, which resulted in 11 power of frequency-band power features per frequency band per epoch and 11 power of the time-domain signal features

Table 6.2: Features extracted from the single-channel EEG signal.

Feature	**Number**	**Purpose**	**Transform**
Power of frequency-band power over the entire epoch	20	Capture the overall presence of the particular frequency band in the signal	$\log(x)$
Power of frequency-band power using a sliding window	220	Capture the presence of the particular frequency band in the signal across time	$\log(x)$
Time-domain signal power over the entire epoch	1	Capture the overall amplitude characteristics of the signal	$\log(x)$
Time-domain signal power using a sliding window	11	Capture the amplitude characteristics of the signal over time	$\log(x)$
Frequency-band power-power correlation	210	Capture the relationships between the different frequency bands over time	None
Time-domain signal autocorrelation	50	Capture long-term dependencies in the signal	x^2
ALL	512		

per epoch. With this sliding window we attempted to capture relationships that are time-specific in particular parts of the epoch, rather than across the whole epoch. This is important, as many sleep scoring patterns, such as sleep spindles and K-complexes, are transient, and are better captured by localising them in time. All the extracted features are summarised in Table 6.2. We mapped all the features in the [0,1] interval, and centred their distribution using transformations (see Table 6.2). We then normalised the features from each trial of each subject.

The AASM manual [49] includes a number of rules that recommend taking into account neighbouring epochs for the scoring of each current epoch under certain circumstances. We identified 12 rules in total concerning the transition between certain sleep stage pairs that refer to 7 distinct transition patterns, as shown in Table 2.3. These rules apply to three sleep stage pairs, N1-N2, N1-R and N2-R. The transition patterns include up to two preceding or succeeding neighbouring epochs. Trying to capture the effect of these transition rules in an automatic sleep scoring algorithm by simply including

transition probabilities between sleep stages is not a suitable approach. The reason is that the algorithm could overfit to hypnogram-level patterns from the subjects we used for training, especially when the training data do not include data from different sleep pathologies.

We incorporated transition information directly as features for our machine learning algorithm. Specifically, for the classification of each epoch, apart from the features corresponding to itself, we included the features from the preceding two and succeeding two epochs. We addressed the possibility of overfitting which exists in this case in our experimental design (Section 6.2). In the literature, Liang *et al.* [62] used 11 *a priori* hypnogram 'smoothing rules' in order to capture transition information. These rules are applied on the scored epochs after automatic sleep scoring has taken place, effectively changing the classification of each epoch given the sleep stage of its neighbours. Unfortunately, the authors described only 2 of the rules in their paper, and, notably, did not discuss the order in which the rules are applied to the estimated hypnogram.

6.1.2 Machine learning methodology

For the classification of the epochs into sleep stages we used a stacked sparse autoencoders model (see Section 4.10). The optimisation method we used was L-BFGS, as recommended in [72]. The hyperparameters of a sparse autoencoder-based model are: (1) a regularisation weight λ which is used to decrease the magnitude of the parameters and prevent overfitting (see Section 4.9), (2) a sparsity weight β which controls the relative importance of the sparsity penalty term (see Section 4.10), (3) a sparsity parameter ρ which sets the desired level of sparsity (see Section 4.10), and (4) the number of units n in the hidden layer of the autoencoder. The only hyperparameter for the optimisation is the total number of iterations r. Our stacked sparse autoencoders model has two hidden layers. Our final choice of hyperparameters was $\lambda = 10^{-5}$, $\beta = 2.0$, $\rho = 0.2$, $n = 20$, and $r = 60$. We used autoencoders with the sigmoid activation function. In the next section, we give the details of how the hyperparameters were chosen.

The classes (sleep stages) in our dataset, as in any PSG dataset, were not balanced, i.e. there were a lot more epochs for some stages (particularly N2) than others (particularly W and N1). In such a situation, if all the data is used as is, it is highly likely that a classifier will exhibit skewed performance favouring the most represented classes, unless the least represented classes are very distinct from the other classes. In order to resolve the issues stemming from imbalanced classes we decided to employ class-balanced random sampling with an ensemble of classifiers, each one being trained on a different sample of the data. Our final model consisted of an ensemble of 20 independent stacked sparse autoencoders (SSAEs) with the same hyperparameters. Each of the 20 SSAEs was trained using a sample of the data in which the number of epochs per stage per recording was equal to the number of epochs of the least represented stage (N1). The classification of the epochs in the testing recordings was done by taking the mean of the class probabilities that each of the 20 SSAEs outputs, and then selecting the class with the highest probability.

We used our own Matlab implementation for time-frequency analysis and stacked autoencoders, and the Matlab implementation by Mark Schmidt for L-BFGS (`http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`).

In terms of the computational burden of our method, on a machine with a processor with 4 cores at 2.8 GHz, 16 GB of RAM and an SSD hard disk, a full run of cross-validation takes approximately 2 hours, or 2 hours $\div$ (20 folds $\times$ 20 models per fold) = 0.005 hours or 18 seconds per single SSAE model.

6.2 Evaluation

To evaluate the generalisability of our method, we obtained our results using 20-fold cross-validation. Specifically, in each fold we used the recordings of a single subject for testing and all other recordings for training. We used each subject's recordings only once for testing, thus obtaining a one-to-one correspondence of cross-validation folds and test subjects. We chose per-subject cross-validation as we also performed comparisons across

individual recordings. With this experimental design, we were able to assess both the overall performance of our method and the performance across recordings with a single set of experimental results.

To choose the hyperparameters we used the training data at each fold of the cross-validation, further splitting it into one training and one validation set for choosing the hyperparameters. This protocol is similar to the double (or *nested*) cross-validation approach [41, pp. 245–7] [8]. In nested cross-validation a second (or *inner*) cross-validation is performed using only the training data at each fold with the objective of optimising the hyperparameters based on the performance of different models having a different choice of hyperparameters. The hyperparameters that produced the model with the highest inner cross-validation performance are chosen. Then a model is built using all of the training data and is tested in the outer cross-validation validation set, that was not seen during hyperparameter optimisation. In our case, it would be very expensive computationally to run a full nested cross-validation, so a single training and validation split within each fold (instead of a full inner cross-validation) was a statistically valid alternative. Concretely, in our work we used a 15-4 training-validation split for hyperparameter optimisation with the 19 subjects that form the training set of each fold.

For choosing the candidate values of hyperparameters grid search is common practice in the deep learning literature [8]. Grid search is the process of presetting possible values for each of the hyperparameters and considering every possible combination of those hyperparameters. However, the combinatorial space to explore all the possible combinations of hyperparameters is very large. This not only creates considerable computational burden for hyperparameter choice, but also could lead to overfitting to the validation set [8],especially in settings similar to our work in which the dataset is not very large. To address these issues we decided to choose the same hyperparameters across all layers, and adopt a greedy process for adding layers to the neural network. Starting with a single-layer neural network we proceeded with adding additional layers only if the addition of a new layer improved performance.

Another consideration in hyperparameter optimisation is that the possible values per hyperparameter that would be worth considering are effectively unknown at the start of training. Doing a grid search across values that may be completely inappropriate would increase the computational burden even more. To address this issue we explored potential values of the hyperparameters across broader ranges only for a couple of folds initially. Once we narrowed down the range of the two or three best values for each hyperparameter using that exploratory approach we proceeded with using all the folds. For example, in the exploratory stage we considered sizes of up to 400 units for the hidden layer of the neural network. After the exploratory stage the effective range was narrowed down to between 20 and 40 units. We should note that we did not evaluate all possible values in the effective hyperparameter range. For example, achieving higher performance with 21 units versus 20 units, although in theory possible would expose us to the possibility of overfitting. Finally, given similar performance with hyperparameters that affect the complexity of the model, we chose the values of these hyperparameters that result in lower model complexity. For example, given a neural network with 3 hidden layers and a network with 2 hidden layers with the same performance, the latter was chosen over the former.

Once the hyperparameters at each fold were chosen in the inner loop, for cross-validation for each hyperparameter we chose the values that are more common across all folds, and assigned those values to all the folds. Necessarily, these values were not the optimal ones across all folds, but this was done to be able to assess the performance of a single model rather than potentially different models (due to hyperparameter variations) across folds. Regarding the sensitivity of our model to the choice of hyperparameters, we observed that adding a third layer or increasing the number of units of each layer from 20 to 40 had a slightly negative effect to performance. Increasing λ from 10^{-5} to 10^{-4}, increasing β from 2.0 to 1.5, or decreasing ρ from 0.2 to 0.1 decreases performance slightly as well. We should note that it is difficult to isolate the effect of individual hyperparameters, as the value of one of them affects the optimal value for another. For

example, there is a close relationship between the number of nodes in each hidden layer and the regularisation weight λ. Specifically, a network with a larger number of nodes may demand a higher regularisation weight.

We report the evaluation metrics using their average across all recordings. Specifically, we report their mean value across all 5 sleep stages and their value for the most misclassified sleep stage, which gives information about the robustness of the method across sleep stages. We tested our method with both available EEG electrodes (F_{pz}-C_z and P_z-O_z) in the Physionet dataset. We report the scoring performance using the best electrode, which was F_{pz}-C_z.

We calculated 95% confidence intervals for each of the performance metrics by bootstrapping using 1000 bootstrap samples across the confusion matrices of the 39 recordings. For each bootstrap sample we sampled the recording indexes (from 1 to 39) with replacement and then added up the confusion matrices of the selected recordings. We then calculated each evaluation metric for each bootstrap sample. We report the mean value of each metric across the bootstrap samples, and the values that define the range of the 95% confidence interval per metric, i.e. the value of the metric in the 26th and 975th position of the ordered bootstrap sample metric values.

We also tested our algorithm using 5-fold cross-validation with non-independent training and testing sets by mixing the subjects' epochs as the authors in [34] did. This was done to show the improvement in the results that such a flawed practice can result into, and appropriately compare our method to [34]. We do not consider this performance indicative of the quality of our method, or any method targeted in EEG sleep scoring, as it is not practical in the real world. These results are separated from the others in Table 7.3.

To further evaluate the generalisability of our method, we performed two tests on our results to assess the correlation between scoring performance and (1) a measure of the sleep quality of each recording, and (2) the percentage of transitional epochs in each recording. Robust scoring performance across sleep quality and temporal sleep variability,

can be seen as further indicators of the generalisability of an automatic sleep stage scoring algorithm. The reason is that low sleep quality and high sleep stage variability across the hypnogram are prevalent in sleep pathologies (see, for example, [74]).

We measured sleep quality with a widely-used index, called *sleep efficiency*. Sleep efficiency is defined as the percentage of the total time in bed that a subject was asleep [88, p. 226]. Our data contain a 'lights out' indicator, which signifies the start of the time in bed. We identified the sleep onset as the first non-W epoch that occurred after lights were out. We identified the end of sleep as the last non-W epoch after sleep onset, as our dataset does not contain a 'lights on' indicator. The number of epochs between the start of time in bed and the end of sleep was the total time in bed, within which we counted the non-W epochs; this was the total time asleep. We defined transitional epochs as those whose preceding or succeeding epochs were of a different sleep stage than them. We computed their percentage with respect to the total time in bed. In our experiments we computed the R^2 and regression coefficient p-value between sleep efficiency and scoring performance, and between percentage of transitional epochs and scoring performance.

The metrics we computed were precision, sensitivity, F_1-score, per-stage accuracy, and overall accuracy.

Finally, we computed the scoring performance of our algorithm without and with features from neighbouring epochs. If we observed improvement in sleep stage pairs which are not included in the transition rules (i.e. any pair other than N1-N2, N1-R and N2-R, see Table 2.3), we would conclude that the algorithm learned spurious patterns that are an artifact of our training data. Additionally, we should observe at least some small improvement and certainly no decrease in the classification performance between pairs N1-N2, N1-R and N2-R. In this case, even without having data from different sleep pathologies we can evaluate whether the epoch-to-epoch or hypnogram-level patterns that our algorithm learned were akin to the generic guidelines or overfitting to the training data.

Table 6.3: Confusion matrix from cross-validation using the F_{pz}-C_z electrode.

	N1 (algorithm)	**N2 (algorithm)**	**N3 (algorithm)**	**R (algorithm)**	**W (algorithm)**
N1 (expert)	**1654** (60%)	**262** (10%)	**8** (0%)	**366** (13%)	**472** (17%)
N2 (expert)	**1270** (7%)	**13696** (78%)	**1231** (7%)	**760** (4%)	**621** (4%)
N3 (expert)	**7** (0%)	**469** (8%)	**4966** (89%)	**6** (0%)	**143** (3%)
R (expert)	**899** (12%)	**340** (4%)	**0** (0%)	**6164** (80%)	**308** (4%)
W (expert)	**441** (13%)	**34** (1%)	**23** (1%)	**138** (4%)	**2744** (81%)

This confusion matrix is the sum of the confusion matrices from each fold. The numbers in bold are numbers of epochs. The numbers in parentheses are the percentage of epochs that belong to the class classified by the expert (rows) that were classified by our algorithm as belonging to the class indicated by the columns.

6.3 Results

As we show in the the normalised confusion matrix in Table 7.2, the most correctly classified sleep stage was N3, with around 90% of stage N3 epochs correctly classified. Stages N2, R and W follow, with around 80% of epochs correctly classified for each stage. The most misclassified stage was N1 with 60% of stage N1 epochs correctly classified. Most misclassifications occurred between the pairs N1-W and N1-R (about 15% and 13% respectively), followed by pairs N1-N2 and N2-N3 (about 8%), and N2-R and R-W (about 4%). The remaining pairs had either misclassification rates smaller than 4% (N2-W and N3-W) or almost no misclassifications at all (N1-N3 and N3-R). We also observe that the percentage of false negatives with respect to each stage (non-diagonal elements in each row) per pair of stages was approximately balanced between the stages in the pair (the only conspicuous exception is the pair N1-W, and, to a lesser extent, the pair N2-W). Effectively the upper and lower triangle of the confusion matrix are close to being mirror images of each other. This is a strong indication that the misclassification errors due to class imbalance have been mitigated.

As we show in Table 7.3, our method has both high overall accuracy (78%, range 75–

80%), and high mean F_1-score (84%, range 82–86%) and mean accuracy across individual sleep stages (86%, range 84–88%) over all subjects. From the scoring performance metrics results in Table 7.3 we observe that our method either outperformed or had approximately equal performance with the methods in the literature in all metrics apart from worst-stage precision (the non-independent testing results at the bottom row are not taken into account). In many cases, even the lower end of the 95% confidence interval (the top number in parentheses) was higher than the corresponding metric for the other methods. Table 7.3 also summarises the improvement of our method over the state of the art, i.e. the best of all the methods in the literature in that particular metric (negative numbers indicate worse performance than the state of the art). Overall, our method exhibits improved performance over the state of the art in automated sleep scoring using single-channel EEG across the five scoring performance metrics.

In Table 6.5 we show the results of the algorithm without and with information from neighbouring epochs. We observe that there is no mutual improvement in any other stages apart from the targeted pairs N1-N2, N1-R and N2-R.

We also assessed the independence of the scoring performance (for F_1-score and overall accuracy) of our method across recordings relative to sleep efficiency and the percentage of transitional epochs per recording (Table 7.4). The p-values of the regression coefficients are all above 0.15, which means that we fail to reject the null hypothesis of zero R^2, which is already negligible (lower than 0.1) in all cases. For clarity we present the data for these tests graphically for the F_1-score results in Figures 6.2 and 6.3. Our dataset contained 10 recordings with sleep efficiency below 90% (in the range 60-89%), which is the threshold recommended in [88, p. 7] for young adults. The percentage of transitional epochs ranged from 10-30% across recordings.

Finally, in Figure 6.4 we present an original manually scored hypnogram and its corresponding estimated sleep hypnogram using our algorithm for a single PSG for which the overall F_1-score was approximately equal to the mean F_1-score across the entire dataset.

Table 6.4: Comparison between our method and the literature across the five scoring performance metrics (precision, sensitivity, F_1-score, per-stage accuracy, and overall accuracy).

	Scoring performance metrics								
	Precision		**Sensitivity**		**F_1-score**		**Accuracy**		
Study	Mean	Worst	Mean	Worst	Mean	Worst	Mean	Worst	Overall
Independent training and testing									
[62]	93	**89**	77	29	82	43	86	63	77
[62]	90	82	73	19	77	31	83	57	73
[11]	92	88	74	36	81	51	84	66	74
	(92)	(86)	(75)	(55)	(82)	(68)	(84)	(74)	(75)
current	**93**	88	**78**	**60**	**84**	**71**	**86**	**76**	**78**
	(94)	(90)	(80)	(65)	(86)	(75)	(88)	(78)	(80)
	0	*-1*	*+1*	*+24*	*+2*	*+20*	*0*	*+10*	*+1*
Non-independent training and testing									
[34]	93	88	77	53	84	68	86	75	77
current	**95**	**91**	**82**	**65**	**88**	**76**	**89**	**79**	**82**
	+2	*+3*	*+5*	*+8*	*+4*	*+8*	*+3*	*+4*	*+5*

For the binary metrics, we report the mean performance (over all five sleep stages) as well as the worst performance (in the most misclassified sleep stage, always stage N1). We present the results for our method using the F_{pz}-C_z electrode with cross-validation using both independent and non-independent training and testing. The numbers in parentheses are the bootstrap 95% confidence interval bounds for the mean performance across subjects. The signed numbers in italics indicate the improvement (positive) or deterioration (negative) in performance over the second best (improvement) or best (deterioration) method in the literature.

6.4 Discussion

Given the high disagreement across epochs between human experts [89] a 1–2% improvement in mean scoring performance may not be considered significant. We think that there are two characteristics that render our method better than the state of the art. First,

Table 6.5: Normalised confusion matrices from 20-fold cross-validation using the F_{pz}-C_z electrode without and with neighbouring epochs. All values are percentages. Pairs of stages with mutual improvement are in bold (N1-N2, N1-R and N2-R).

	Algorithm									
	Without neighbouring epochs					**With neighbouring epochs**				
	N1	**N2**	**N3**	**R**	**W**	**N1**	**N2**	**N3**	**R**	**W**
N1 (expert)	53	**11**	0	**17**	18	60	**9**	0	**13**	17
N2 (expert)	**8**	77	7	**5**	4	**7**	78	7	**4**	4
N3 (expert)	0	8	89	0	3	0	8	89	0	3
R (expert)	**18**	**5**	0	73	5	**12**	**4**	0	80	4
W (expert)	13	1	1	4	82	13	1	1	4	81

Table 6.6: Correlation between sleep efficiency and percentage of transitional epochs, and scoring performance (F_1-score and overall accuracy).

	Recording parameters			
	Sleep efficiency		**Percentage of transitional epochs**	
Metric	R^2	p-value	R^2	p-value
F_1-score	0.02	0.42	0.04	0.20
Overall accuracy	0.02	0.46	0.05	0.17

we significantly decreased the gap between the mean performance over all sleep stages and the most misclassified stage performance (stage N1) compared to the state of the art with about 20% improvement in the F_1-score and 10% improvement in accuracy over the state of the art (with independent testing). Second, we mitigated the adverse effects of class imbalance to sleep stage scoring. This is an indication that our method could be generalised to data with varying proportions across sleep stages, and is not markedly affected by these proportions, as other methods in the literature seem to be by inspecting their normalised confusion matrices. After addressing class imbalance, the majority of the remaining misclassification errors is likely due to either differences in EEG patterns that our feature extraction methodology cannot sufficiently capture, difficulty in captur-

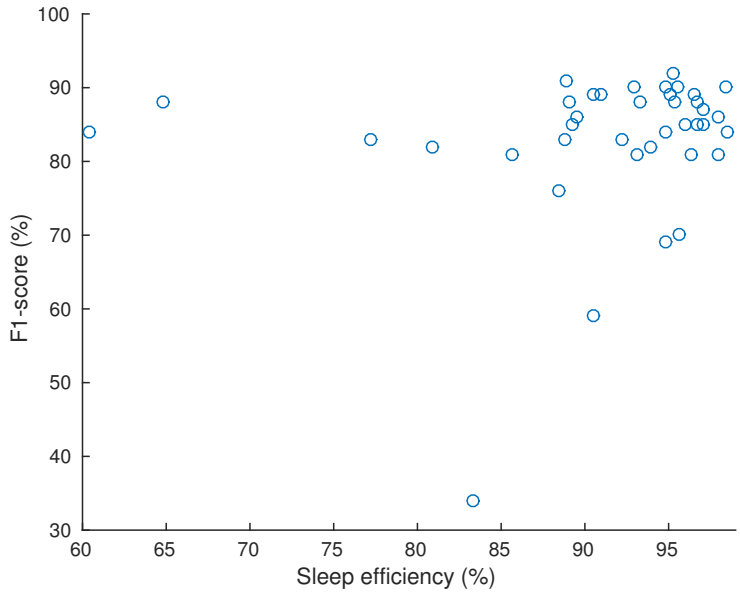

Figure 6.2: F_1-score as a function of sleep efficiency.

ing EOG and EMG-related that are important in distinguishing between certain sleep stage pairs features through the single channel of EEG, or inherent similarities between sleep stages in epochs that even experts would disagree with one another about.

The most misclassified pair of sleep stages using our method was N1-W; about 15% false negatives for each stage were accounted for by the other. We think that the root cause of the problem is the similarity in the characteristic EEG frequency patterns of sleep stages N1 and W, as described in the AASM sleep scoring manual [49]. Specifically, relatively low voltage mixed 2–7 Hz and alpha (8–13 Hz) activity are described as criteria for both stages. The second most misclassified pair of sleep stages was N1-R, for which the characteristic EEG frequency patterns are similar as well. There are 4 transition rules which pertain to the N1-R pair in the AASM manual, which have proven useful, as we showed in Table 6.5. However, some of these rules rely heavily on EOG and EMG, so it was difficult to exploit their full potential. The next most misclassified pairs of sleep stages were N1-N2 and N2-N3 (about 8%). The classification between stages N1 and N2 depends to a great extent on transition patterns (Table 2.3) that partly rely on the detection of arousals (and, in particular, on K-complexes associated or not with

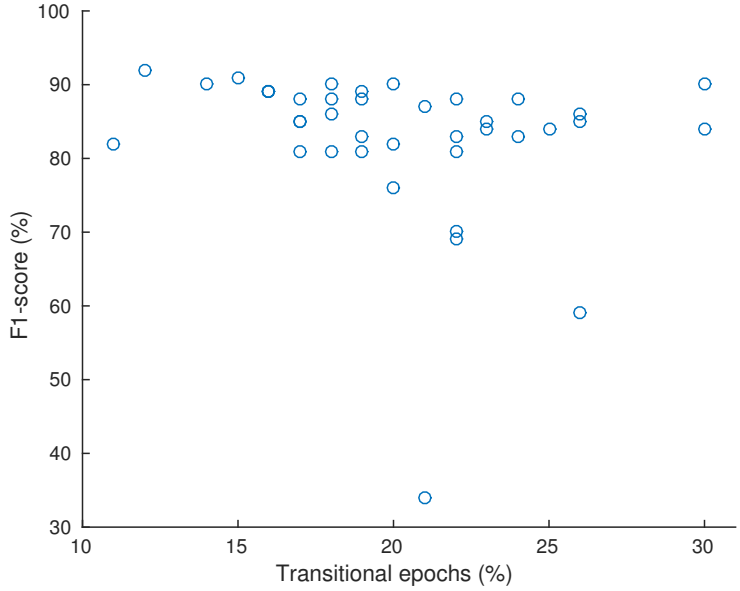

Figure 6.3: F_1-score as a function of transitional epochs.

arousals), body movements and slow eye movements, which can be difficult to capture using a single channel of EEG. The misclassification between stages N2 and N3 could be partly attributed to the potential persistence of sleep spindles in stage N3 [49, p. 27].

Of the two electrodes in the dataset, we achieved better results using the signal from electrode F_{pz}-C_z. We hypothesised that this was due to fact that the F_{pz}-C_z position can better capture most of the frequency band activity that is important for sleep staging. Specifically, delta activity [32], K-complexes [40] and lower frequency sleep spindles [51] are predominantly frontal phenomena, and alpha activity, although it is predominantly an occipital phenomenon, can manifest itself in frontal derivations [32]. Theta activity [32] and higher frequency sleep spindles [51] are mostly parietal phenomena. However, theta activity is present in multiple sleep stages, so even if it were captured more effectively from the P_z-O_z position it might not have been very beneficial by itself.

Although we recognise that our dataset does not contain a very large number of recordings of bad sleep quality, we found no statistically significant correlation between sleep efficiency and mean scoring performance. Similarly, there was no statistically significant correlation between the percentage of transitional epochs (which are by definition more

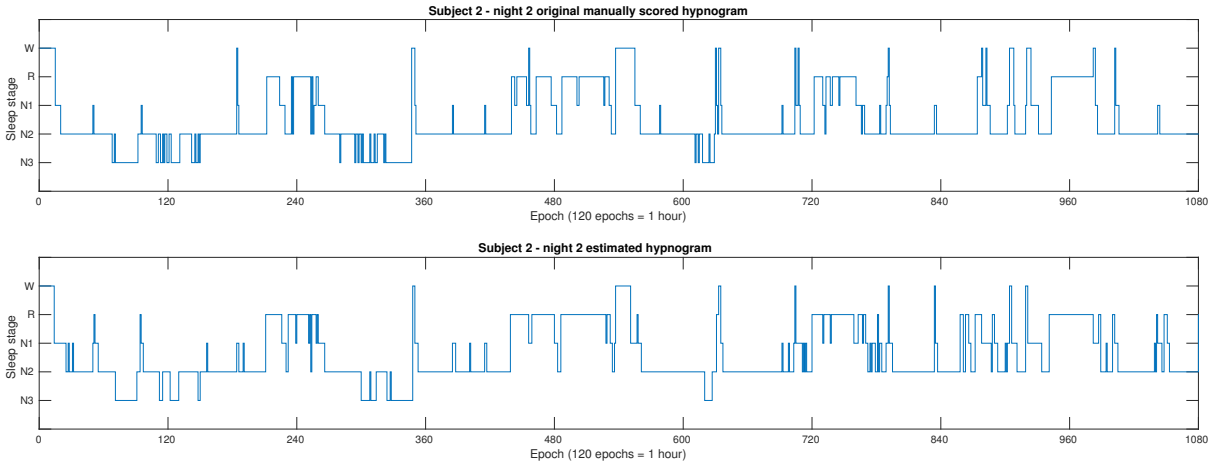

Figure 6.4: The original manually scored hypnogram (top) and the estimated hypnogram using our algorithm (bottom) for the second night of subject number 2.

ambiguous) and mean sleep scoring performance. These statistical test results indicate that our method could be robust across a number of potentially adverse factors.

Mean interrater agreement between human sleep scorers across subjects and stages can vary significantly. For example, in [89] the consensus agreement among three experts was between 60-80%. It would therefore be desirable that the difference in the performance of an automated scoring algorithm across scorers is not significant (i.e. that the algorithm does not overfit to a specific expert's scoring style). Each recording in our dataset was scored by one of six different experts. In total there are 27 recordings scored by a single expert, and 12 recordings scored by all other five experts combined. The number of recordings per expert was not sufficiently large to perform a formal statistical test to assess the significance of differences in scoring performance across experts. Both the mean F_1-score for the recordings scored by the single expert and the mean F_1-score for the recordings scored by any of the other experts were between 83–84%. Both values are close to each other and the overall F_1-score.

For different pathologies that are related with sleep disorders, there are different sleep stages that are relatively more important for distinguishing them from normal sleep. For instance, to distinguish normal sleep from sleep in patients with depression stages R and N3 are relatively more important than other stages (see for example [83]). Common

measures of sleep quality, include sleep efficiency, wake after sleep onset and sleep latency [88, p. 226], for all of which detection of stage W is essential. Different drugs are associated with effects in all non-R sleep stages N1, N2 and N3 [88, p. 9]. Excessive daytime sleepiness and sudden-onset sleep (sudden W to N2 transition) are present in Parkinson's disease [43], and detection of stages N1 and N2 are particularly important for those. These examples indicate the broad range of sleep architecture aspects that need to be targeted across different pathologies. Therefore, the accurate scoring of the entire sleep architecture would be beneficial for a wide range of biomedical applications.

Our method can account for case-specific relative importance of sleep stages in a straightforward way. Our classification algorithm outputs class probabilities. Since in this chapter we placed the same weight to each sleep stage, we classified each epoch to the stage that had the highest class probability. If we wanted to place different weight to each class, we could multiply each stage's probability with a stage-specific weight before choosing the stage with the highest class probability (of course, these weights should be the same for each classified epoch). This would incorporate the relative importance that a researcher places on each sleep stage given the specific sleep pathology that they are trying to identify.

To the best of our knowledge our method has the best performance in the literature when classification is done across all five sleep stages simultaneously using a single channel of EEG. This is different from doing fewer than five one-vs-all classification tasks, as in the latter case, if the eventual overall objective is simultaneous 5-class classification, the performance is likely overestimated. There are examples in the literature that achieve higher performance in a single or two one-vs-all classification tasks, especially for the most easily distinguishable stages N3 and W. However, this is not the same as achieving high performance in a 5-class classification problem, because the errors in the remaining classes are not taken into account. Therefore, since our method achieved very high performance for stages N3 and W, while *simultaneously* achieving good performance in the remaining stages, it is preferable to a method that achieves high performance in a stage W versus

N3-only classification task.

Chapter 7

Automated sleep scoring using convolutional neural networks

In this chapter we present another approach for automatic sleep stage scoring using single-channel EEG [95]. We used CNNs to learn task-specific filters for classification without using prior domain knowledge, as we did in Chapter 6. We used class-balanced random sampling within the stochastic gradient descent (SGD) optimization of the CNN to avoid skewed performance in favour of the most represented sleep stages. We achieved high mean F_1-score (79%, range 81–83%), mean accuracy across individual sleep stages (80%, range 82–84%) and overall accuracy (74%, range 71–76%) over all subjects. By analyzing and visualizing the filters that our CNN learns, we show that for the task of sleep stage scoring the learned filters qualitatively correspond to the sleep scoring criteria in the American Academy of Sleep Medicine (AASM) manual that human experts follow. Our results are comparable to state-of-the-art methods with hand-engineered features, and our method's performance is balanced across classes. We show that without using prior domain knowledge a CNN can automatically learn the distinguishing characteristics among different sleep stages.

7.1 Convolutional neural network architecture

In our CNN architecture we are using the raw EEG signal without preprocessing as the input. Using raw input (usually with some preprocessing) in CNN architectures is the norm in applications of deep learning in computer vision. In classification problems with one-dimensional (1D) signals CNNs can also be applied to a precomputed spectrogram or other time-frequency decomposition of the signal, so that the input to the CNN is a two-dimensional (2D) stack of frequency-specific activity over time. Characteristic examples of this approach can be found in recent work in signal processing for speech and acoustics [86, 28, 45, 102]. When the spectrogram is used as input it can be treated as a 2D image. Recently, there has been also growing interest in applying CNNs to raw 1D signals. Again, there are characteristic examples from speech and acoustics in [28, 77, 90, 78, 44].

Our CNN architecture, shown in Figure 7.1, comprises two pairs of convolutional and pooling layers (C1-P1 and C2-P2), two fully connected layers (F1 and F2), and a softmax layer. Between layer P1 and layer C2, we include a 'stacking layer', S1. As shown in Table 7.1, layer C1 contains 20 filters, so that the output of layer C1 is 20 filtered versions of the original input signal. These filtered signals are then subsampled in layer P1. The stacking layer rearranges the output of the layer P1, so that instead of 20 distinct signals the input to the next convolutional layer C2 is a 2D stack of filtered and subsampled signals. As shown in Figure 7.1 and Table 7.1, the filters in layer C2 are 2D filters. The height of the layer C2 filters is 20, same as the height of the stack. The purpose of these 2D filters is to capture relationships across the filtered signals produced by filtering the original signal in layer C1, across a specific time window.

With this CNN architecture we attempt to combine a CNN architecture using raw signals [66, 28, 77, 90, 78, 44] with the idea of using a 2D stack of frequency-specific activity over time [68, 86, 28, 45, 102]. In a standard CNN architecture layer C2 would have the same structure as layer C1, with a number of 1D filters applied to each of layer P1 outputs. The most common way to combine information across the layer P2

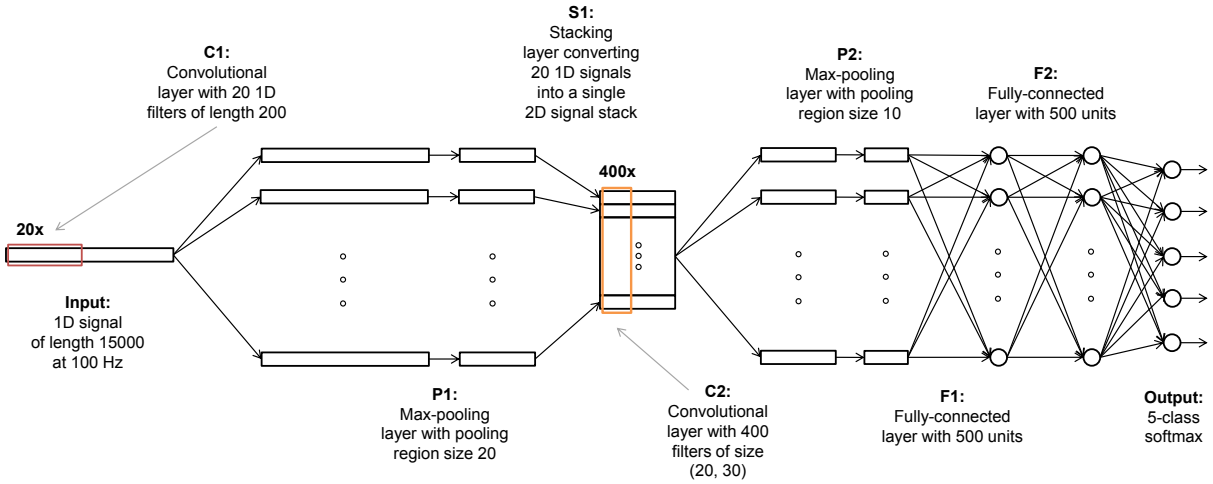

Figure 7.1: CNN architecture

outputs is by adding up the filtered signals of layer C2 across layer P2 outputs by filter index [61]. While this has an effect similar to the stacking layer, we think that explicitly stacking the outputs of layer P2 makes clear the correspondence between CNN methods and hand-engineered feature methodologies.

The cost function for the training of our CNN architecture was the softmax with l^2 regularization. We applied the rectified linear unit (ReLU) nonlinearity after convolution and before pooling. The hyperparameters of a CNN are: the number and types of layers, the size of the filters for convolution and the convolution stride for each convolutional layer, the pooling region size and the pooling stride for each pooling layer, and the number of units for each fully connected layer. We summarise the selected hyperparameters for our CNN architecture in Table 7.1.

In order to resolve the issues stemming from imbalanced classes, in Chapter 6 we employed class-balanced random sampling with an ensemble of 20 classifiers, each one being trained on a different balanced sample of the data. This is not an efficient way for class-balancing with CNNs, as training even a single CNN is very time-consuming. The strategy that we followed in this chapter was different. At each epoch of SGD we used a different class-balanced batch for the optimization.

As shown in Table 2.3 the scoring of a particular epoch can depend on the characteristics of the preceding or succeeding epochs, for the sleep stage pairs N1-N2, N1-R, and

Table 7.1: The CNN architecture that we used in this chapter. It has two pairs of convolution and pooling layers, one stacking layer between the two pairs, two fully connected layers, and one softmax layer for classification.

Layer	Layer Type	# Units	Unit Type	Size	Stride	Output Size
Input						(1, 1, 15000)
C1	convolutional	20	ReLU	(1, 200)	(1, 1)	(20, 1, 14801)
P1	max-pooling			(1, 20)	(1, 10)	(20, 1, 1479)
S1	stacking					(1, 20, 1479)
C2	convolutional	400	ReLU	(20, 30)	(1, 1)	(400, 1, 1450)
P2	max-pooling			(1, 10)	(1, 2)	(400, 1, 721)
F1	fully connected	500	ReLU			500
F2	fully connected	500	ReLU			500
Output	softmax	5	logistic			5

N2-R. Therefore, we chose the input data to our CNN to be the signal of the current epoch to be classified together with the signals of the preceding two and succeeding two epochs, as a single, continuous signal, starting from the earliest epoch, with the current epoch in the middle. At the sampling rate of 100 Hz this gives an input size of 15,000 timepoints.

We implemented the CNN architecture using the Python libraries Lasagne (`https://github.com/Lasagne/Lasagne`) and Theano (`https://github.com/Theano/Theano`).

In terms of the computational burden of our method, on a machine with a processor with 4 cores at 2.8 GHz, 16 GB of RAM and an SSD hard disk, each fold of the cross-validation takes approximately 5 hours to run, and therefore a full run of cross-validation takes approximately 100 hours or 4 days for all 20 folds.

7.2 Evaluation

To evaluate the generalizability of the algorithms, we obtained our results using 20-fold cross-validation as in Chapter 6. Specifically, in each fold we use the recordings of a

single subject for testing and the recordings of the remaining 19 subjects for training and validation. For each fold we used the recordings from 4 randomly selected subjects as validation data and the recordings from the remaining 15 subjects for training. The classification performance in the validation data was used for choosing the hyperparameters and as a stopping criterion for training to avoid overfitting to the training data.

The metrics we computed were precision, sensitivity, F_1-score, per-stage accuracy, and overall accuracy. We report the evaluation metrics across all folds. Specifically, we report their mean value across all 5 sleep stages and their value for the most misclassified sleep stage, which provides information about the robustness of the method across sleep stages. We tested our method with the F_{pz}-C_z electrode, with which we had achieved better performance in Chapter 6.

We calculated 95% confidence intervals for each of the performance metrics by bootstrapping using 1000 bootstrap samples across the confusion matrices of the 39 recordings. To further evaluate the generalisability of our method, we performed two tests on our results to assess the correlation between scoring performance and (1) a measure of the sleep quality of each recording, and (2) the percentage of transitional epochs in each recording. We computed the R^2 and the regression coefficient p-value between sleep efficiency and scoring performance, and between percentage of transitional epochs and scoring performance.

We compared our CNN results with our previous work from Chapter 6, as well as a CNN architecture that uses the Morlet wavelets from Chapter 6 to produce a time-frequency stack that is fed to the CNN from the second convolutional layer C2 onwards.

7.3 CNN filter analysis and visualisation

Apart from performance evaluation an additional type of evaluation is required when using CNNs, in our view. As the filters in CNNs are automatically learned from the training data, we need to evaluate whether the filters learned in different folds (i.e. using

different training data) are similar across folds. We analyzed and compared the learned filters from the first convolutional layer of the CNN from each of the 20 different folds. For all of the architectures layer C1 has 20 filters. We extracted the frequency content of the filters by computing the power at different frequency bands using the Fourier transform.

We then fed the testing data for that fold to the CNN. We extracted the features produced by each filter per training example for the middle segment of the signal (the current epoch). Each feature is a signal which represents the presence of the filter over time. We computed the power of the feature signal for each testing example, and then took the mean power across all testing examples of each true (not predicted) class. Some filters have naturally lower power, because they correspond to patterns localized in time and not in continuous activity as shown in the scoring criteria in Table 2.2.

We observed that certain sleep stages produce higher filter activations across all filters in general. To account for those differences, we normalized (to unit length) the power first by sleep stage across filters, and then by filter across sleep stages. Similar filters learned in each fold are generally not at the same index. For easier visual inspection of the results we ordered the filters by the sleep stage for which they have the greatest mean activation (Figures 7.6, 7.7, 7.8 and 7.9).

Finally, we qualitatively compared the learned filters with the guidelines in the AASM sleep scoring manual. To do so we also compared the filters and activation patterns per filter per sleep stage with the frequency content and activation patterns of the hand-engineered Morlet wavelets we used in Chapter 6.

7.4 Results

7.4.1 Sleep stage scoring performance

As we show in the normalized confusion matrix in Table 7.2, the most correctly classified sleep stage was N3 with around 90% of stage N3 epochs correctly classified. Stages R and N2 follow with around 75% of epochs correctly classified for each stage. Stage W has

around 70% of epochs correctly classified. The most misclassified sleep stage was N1 with 60% of stage N1 epochs correctly classified. Most misclassifications occurred between the pairs N1-W and N1-R (about 15%), followed by pairs N1-N2, N2-R and N2-N3 (about 8%), and R-W and N2-W (about 5%). The remaining pairs, N1-N3, N3-R and N3-W have misclassification rates close to zero.

The percentage of false negatives with respect to each stage (non-diagonal elements in each row) per pair of stages was approximately balanced between the stages in the pair. An exception is the pair N1-W, which appears slightly skewed (3% difference) in favor of stage N1. Effectively the upper and lower triangle of the confusion matrix are close to being mirror images of each other. Similarly to Chapter 6, this is a strong indication that the misclassification errors due to class imbalance have been mitigated.

As we show in Table 7.3, our method has high mean F_1-score (79%, range 81–83%), mean accuracy across individual sleep stages (80%, range 82–84%) and overall accuracy (74%, range 71–76%) over all subjects. From the scoring performance metrics results in Table 7.3 we observe that our method has slightly worse performance than our previous work from Chapter 6 [94]. We should note though that the 95% confidence intervals overlap for the majority of the metrics (worst-stage precision, mean and worst-stage sensitivity, mean and worst-stage F_1-score, and worst-stage and overall accuracy), and are otherwise nearly overlapping for the remaining metrics (mean precision and mean accuracy).

We also assessed the independence of the scoring performance (for F_1-score and overall accuracy) of our method across recordings relative to sleep efficiency and the percentage of transitional epochs per recording (Table 7.4). The p-values of the regression coefficients are all above 0.25. The R^2 is already negligible (lower than 0.05) in all cases. For clarity we present the data for these tests graphically for the F_1-score results in Figures 7.2 and 7.3. Our dataset contained 10 recordings with sleep efficiency below 90% (in the range 60-89%), which is the threshold recommended in [88, p. 7] for young adults. The percentage of transitional epochs ranged from 10-30% across recordings.

Table 7.2: Confusion matrix from cross-validation using the F_{pz}-C_z electrode.

	N1 (algorithm)	**N2 (algorithm)**	**N3 (algorithm)**	**R (algorithm)**	**W (algorithm)**
N1 (expert)	**1657** (60%)	**259** (9%)	**9** (0%)	**427** (15%)	**410** (15%)
N2 (expert)	**1534** (9%)	**12858** (73%)	**1263** (7%)	**1257** (7%)	**666** (4%)
N3 (expert)	**9** (0%)	**399** (7%)	**5097** (91%)	**1** (0%)	**85** (2%)
R (expert)	**1019** (13%)	**643** (8%)	**3** (0%)	**5686** (74%)	**360** (5%)
W (expert)	**605** (18%)	**171** (5%)	**47** (1%)	**175** (5%)	**2382** (70%)

This confusion matrix is the sum of the confusion matrices from each fold. The numbers in bold are numbers of epochs. The numbers in parentheses are the percentage of epochs that belong to the class classified by the expert (rows) that were classified by our algorithm as belonging to the class indicated by the columns.

Finally, in Figure 7.4 we present an original manually scored hypnogram and its corresponding estimated sleep hypnogram using our algorithm for a single PSG for which the overall F_1-score was approximately equal to the mean F_1-score across the entire dataset.

7.4.2 CNN filter analysis and visualisation

We computed the frequency content and mean activation per sleep stage for the hand-engineered Morlet wavelet filters in Chapter 6 [94] as a reference. This visualisation is shown in Figure 7.5. In Figures 7.6, 7.7, 7.8 and 7.9 we show the filter visualisation for all 20 folds of the cross-validation. This allows us to observe patterns of similarity between the filters learned using different subsets of subjects for training.

Our general observation is that the filters learned by the CNNs at different folds exhibit certain high-level similarities which are consistent across folds. We observed that filters with highest power in frequencies 1-1.5 Hz usually combined with 12.5-14 Hz are associated with highest activation in stage N3 epochs. Filters with highest power in frequencies 13-14.5 Hz usually combined with 2-4 Hz, are associated with highest

Table 7.3: Comparison between our CNN method [95] and our state-of-the-art results with hand-engineered features (Chapter 6, [94]) on the same dataset across the five scoring performance metrics (precision, sensitivity, F_1-score, per-stage accuracy, and overall accuracy).

	Scoring performance metrics								
	Precision		**Sensitivity**		**F_1-score**		**Accuracy**		
Study	Mean	Worst	Mean	Worst	Mean	Worst	Mean	Worst	Overall
	(92)	(86)	(75)	(55)	(82)	(68)	(84)	(74)	(75)
Ch. 6 [94]	**93**	**88**	**78**	**60**	**84**	**71**	**86**	**76**	**78**
	(94)	(90)	(80)	(65)	(86)	(75)	(88)	(78)	(80)
CNN with	(90)	(82)	(71)	(48)	(79)	(61)	(80)	(67)	(71)
Morlet	**91**	**85**	**73**	**52**	**81**	**64**	**81**	**69**	**73**
wavelets	(92)	(87)	(75)	(56)	(83)	(68)	(83)	(72)	(75)
	(90)	(84)	(71)	(53)	(79)	(66)	(80)	(70)	(71)
CNN [95]	**91**	**86**	**74**	**60**	**81**	**70**	**82**	**73**	**74**
	(92)	(88)	(76)	(66)	(83)	(75)	(84)	(76)	(76)

For the binary metrics, we report the mean performance (over all five sleep stages) as well as the worst performance (in the most misclassified sleep stage, always stage N1). We present the results for our method using the F_{pz}-C_z electrode with 20-fold cross-validation. The numbers in parentheses are the bootstrap 95% confidence interval bounds for the mean performance across subjects. The numbers in bold are the mean metrics values from bootstrap.

Table 7.4: R^2 between sleep efficiency and percentage of transitional epochs, and scoring performance (F_1-score and overall accuracy).

	Recording parameters			
	Sleep efficiency		**Percentage of transitional epochs**	
Metric	R^2	p-value	R^2	p-value
F_1-score	0.04	0.25	0.01	0.50
Overall accuracy	0.03	0.30	0.01	0.55

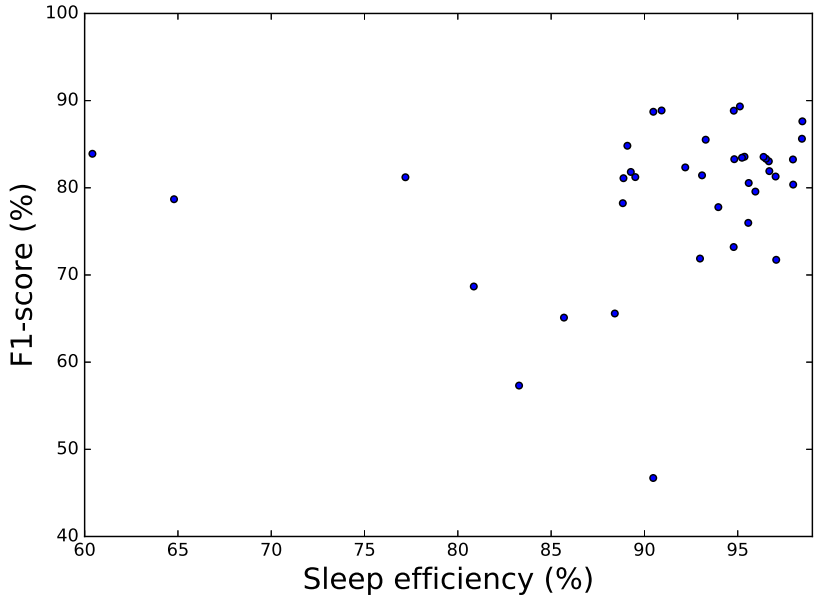

Figure 7.2: F_1-score as a function of sleep efficiency.

activation in stage N2 epochs. High power below 1 Hz filters are associated with highest activation in stage W epochs. Filters with highest power in frequencies 2-5 Hz mostly combined with 14 Hz are associated with highest activation in stage R epochs. It is worth mentioning that the 2-5/14 Hz filters associated with stage R do not contain frequencies from 20-50 Hz. Stage N1 is commonly associated in the majority of folds with filters combining frequencies of 7 Hz and 9 Hz (but not 8 Hz), and always contain frequencies from 20-50 Hz. A common characteristic of all the CNN filters across folds is the absence of filters with frequencies from 10.5-12 Hz and from 15-17 Hz.

7.5 Discussion

In Table 7.3 we compare the performance of our method with hand-engineered features and stacked sparse autoencoders from Chapter 6 [94] (SAE model), our CNN model, and an 'intermediate' model which is using the hand-engineered Morlet wavelets of [94] (shown in Figure 7.5) as the first fixed (i.e. *untrainable*) layer of the CNN (M-CNN model) shown in Figure 7.1. We should note that the architecture used for the M-CNN

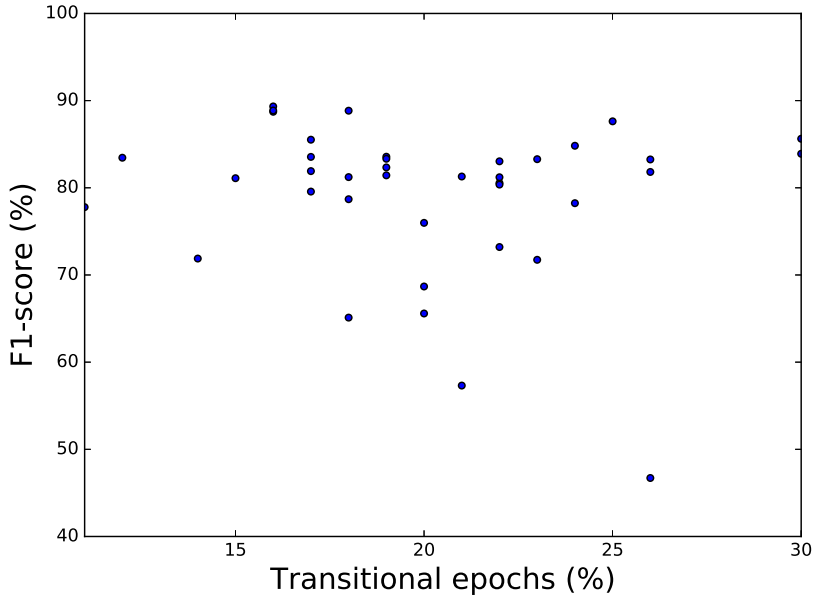

Figure 7.3: F_1-score as a function of transitional epochs.

model was not optimized for the fixed filters, but is exactly the same as the CNN model, to allow us to assess the effect that fixing the filters in the first layer of our CNN model has.

The overall picture that arises from inspecting Table 7.3 is that in terms of performance the SAE model outperforms the CNN model, which in turn outperforms the M-CNN model. We believe that the reason for that is that the CNN model would require a lot more data to achieve performance improvements over the SAE model. Moreover, the SAE model is an *ensemble* of 20 models, whereas the CNN model is a single model. As the required training time of a single CNN model is significantly longer than the training time of a single SAE model (more than 20 times longer) it was not possible to train an ensemble of CNN models, which could improve the performance. We believe that the reason the M-CNN model performs worse than the CNN model is that, by its nature, the M-CNN model works with predefined features for the input, and, consequently, does not optimise the first layer with respect to the classification problem, therefore not fully exploiting the capabilities of a CNN architecture.

Worst-stage performance over all metrics is much closer between the SAE and the

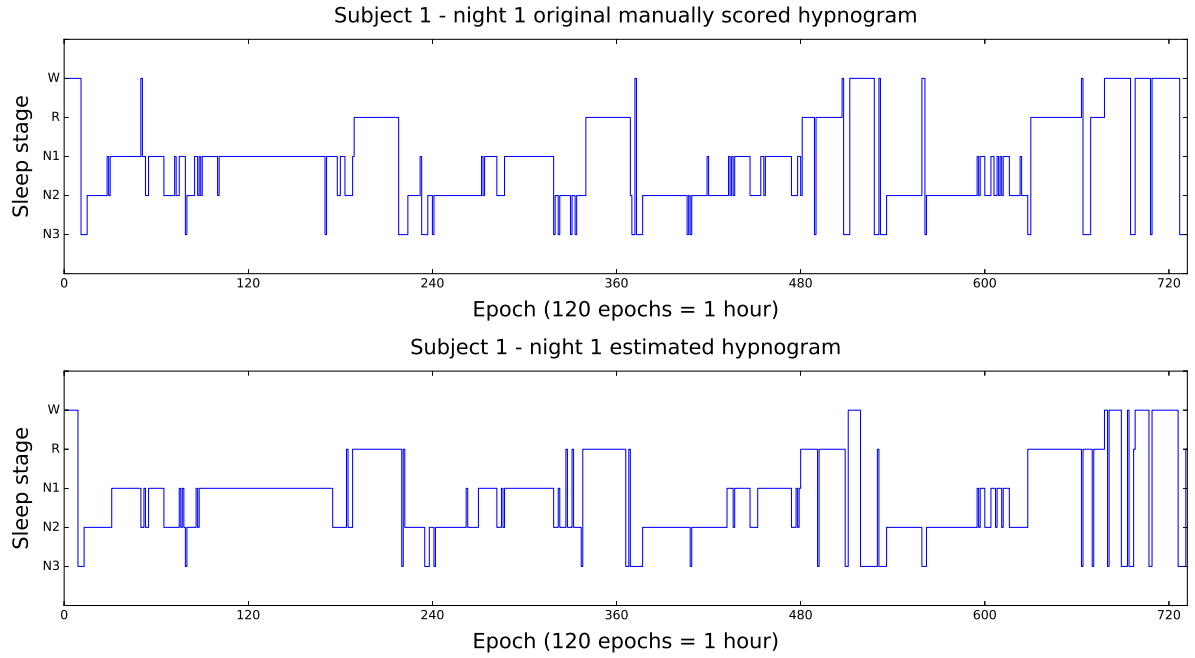

Figure 7.4: The original manually scored hypnogram (top) and the estimated hypnogram
using our algorithm (bottom) for the first night of subject 1.

CNN model compared to mean performance, in which the SAE model is 3-4% better. Mean performance is almost identical between the CNN and the M-CNN model. However, for the M-CNN model worst-stage performance is much lower than either the SAE or the CNN model. However, we observe that the 95% confidence intervals across subjects overlap across the three models, across almost all of the metrics (the two exceptions are mean and worst-stage accuracy between the SAE and the M-CNN model). This indicates

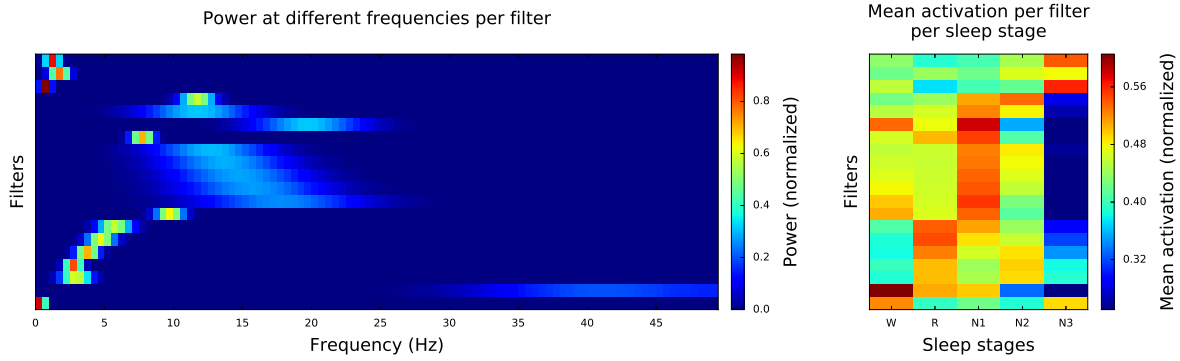

Figure 7.5: Filter visualisation for the hand-engineered filters from [94].

that the differences in performance across subjects between the SAE and the CNN model are not statistically significant.

From the results in Table 7.3 we observe two broad points. The first is that hand-engineering of features based on the AASM manual (SAE model) has better performance than automatic filter learning (CNN model), but the difference based on the dataset we used does not appear to be statistically significant. Using a larger dataset could help clarify the differences in performance between the two models. In general, we expect that a larger dataset would be beneficial for the performance of the CNN model, as CNN models can be difficult to train effectively with smaller datasets. The second point from Table 7.3 is that using a fixed set of filters for the first CNN layer (M-CNN model) achieves worse performance than an end-to-end CNN (CNN model). However, the differences between the two models do not appear to be statistically significant.

Similarly to our previous work [94] the CNN model exhibits balanced sleep scoring performance across sleep stages. The majority of misclassification errors is likely due especially to EOG and EMG-related patterns that are important in distinguishing between certain sleep stage pairs (see Tables 2.2 and 2.3), which are difficult to capture through the single channel of EEG. We experimented with a number of filters larger than 20, but our results did not improve, and, in some cases, deteriorated. This corroborates our hypothesis that the remaining misclassification errors are due to difficulty in capturing patterns from other modalities of the PSG.

Although we recognize that our dataset does not contain a very large number of recordings of bad sleep quality, we found no statistically significant correlation between sleep efficiency and mean scoring performance (see Table 7.4 and Figure 7.2). Similarly, there was no statistically significant correlation between the percentage of transitional epochs (which are by definition more ambiguous) and mean sleep scoring performance (see Table 7.4 and Figure 7.3).

Turning to the filters that are learned by the CNN model, we observe that, in general, the first layer filters that our CNN architecture learns are consistent with the AASM

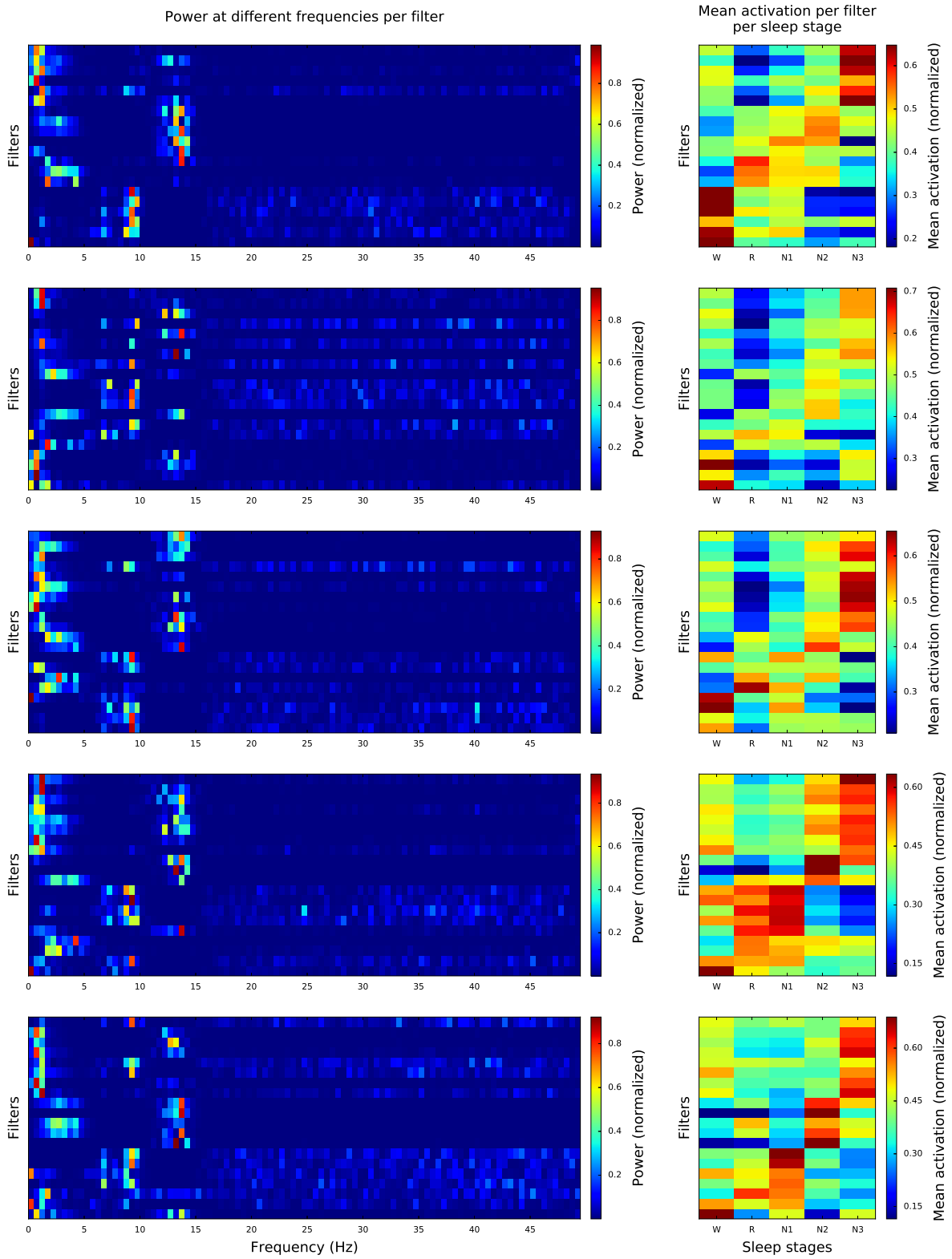

Figure 7.6: Filter visualisation for folds 1 to 5.

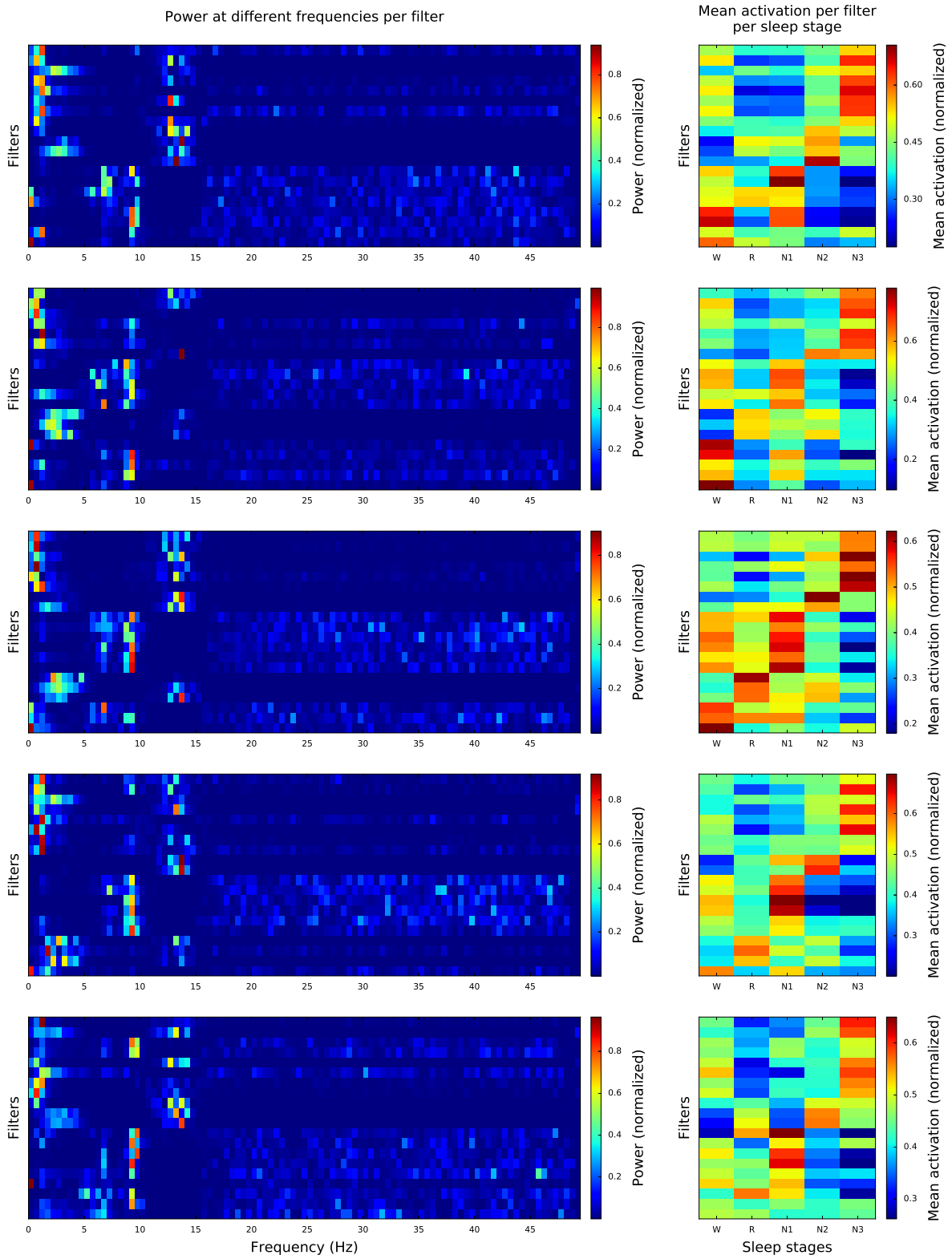

Figure 7.7: Filter visualisation for folds 6 to 10.

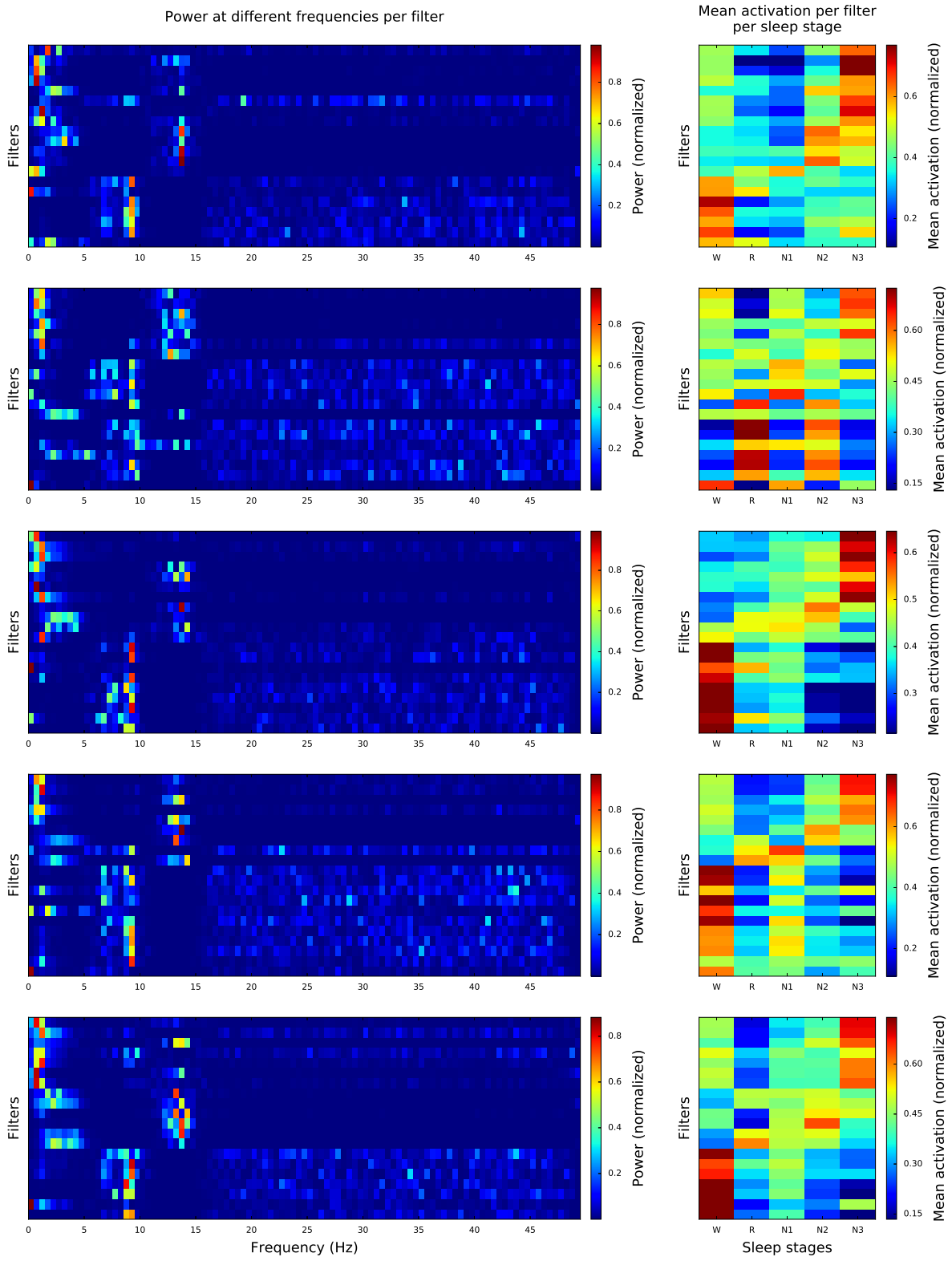

Figure 7.8: Filter visualisation for folds 11 to 15.

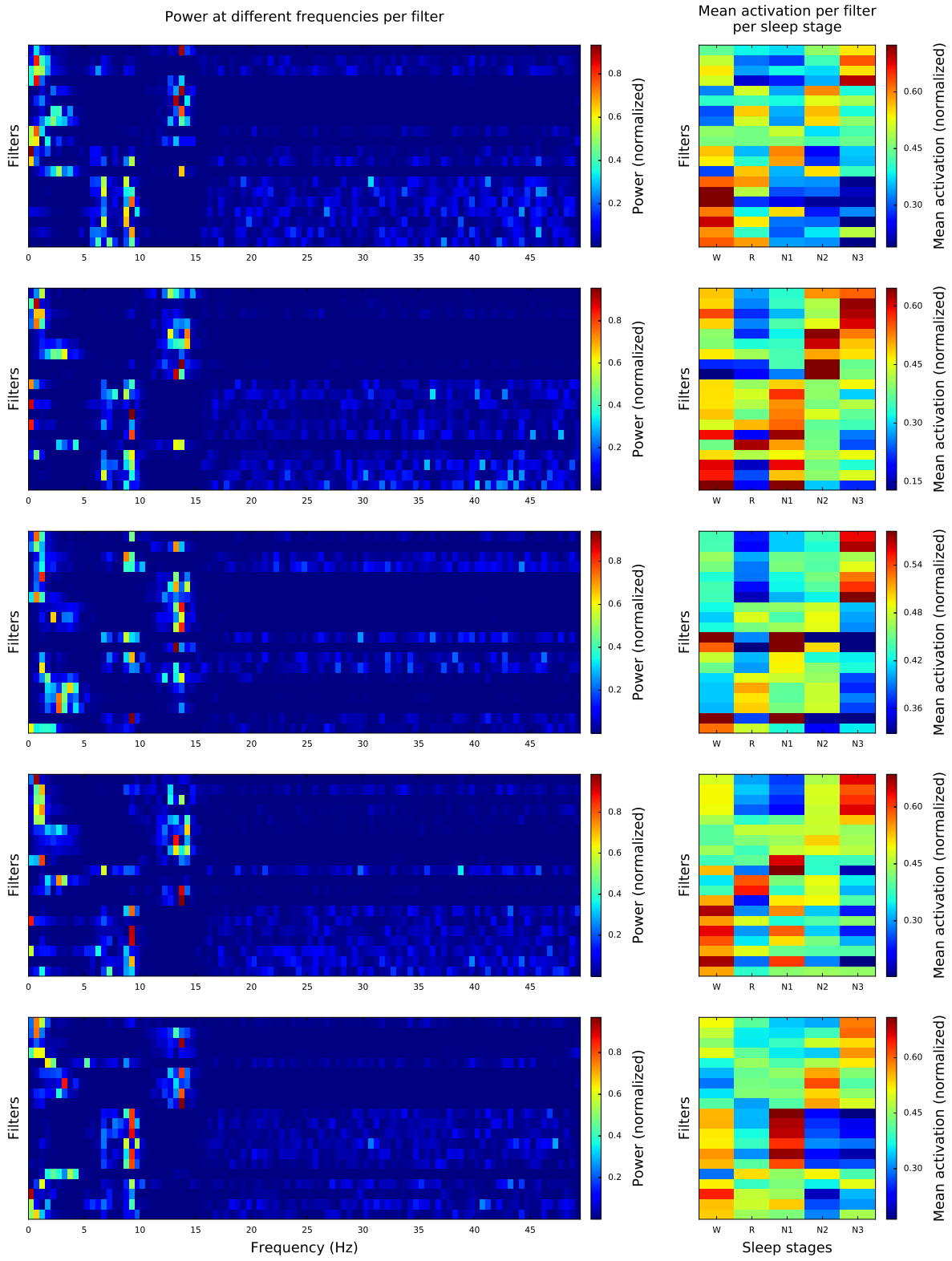

Figure 7.9: Filter visualisation for folds 16 to 20.

sleep scoring manual's guidelines (see Figures 7.6, 7.7, 7.8 and 7.9). The first instance of consistency with the AASM sleep scoring manual are the 1-1.5 Hz and 1-1.5/12.5-14 Hz filters associated with stage N3 epochs. As shown in Table 2.2 stage N3 is associated with activity <2 Hz. Interestingly, activity in 12.5-14 Hz is associated with sleep spindles, a characteristic pattern of stage N2, that can however potentially persist to stage N3 [49, p. 27], and whose interplay with slow wave activity has been associated with memory processing [69]. These filters exhibit little to no activation for stage N2. Therefore, it appears that the CNN learns that there are certain stage N3 epochs in which sleep spindles persist. The second instance of consistency with the AASM manual is the 13-14.5 Hz and 13-14.5/2-4 Hz filters associated with stage N2 epochs. Clearly, the 13-14.5 Hz filters are learned to capture sleep spindles, and the 13-14.5/2-4 Hz filters add K complexes (2-4 Hz activity). Interestingly, K complexes are known to be commonly followed by sleep spindles. In response to that, the CNN does not learn separate 2-4 Hz filters, but only filters that combine the detection of K complexes with the detection of subsequent sleep spindles. We think that this particular specialization of 1-1.5/12.5-14 Hz (stage N3) and 13-14.5/2-4 Hz (stage N2) filters is an indication of the power of CNNs. Incorporating such patterns into a hand-engineered features approach would demand both extensive prior knowledge as well as time-consuming filter design, while with a CNN these patterns are learned directly from the data.

The CNN also learns filters that are consistent across folds for stage R epochs. But while 2-5 Hz activity is clearly described in the AASM manual, these filters consistently exhibit two other characteristics: activity around 14 Hz, and *absence* of activity in high frequencies (20-50 Hz). It is instructive to examine these characteristics in conjunction with the filters for stage N1 epochs, since stages N1 and R are frequently confused with one another, as shown in the confusion matrix of Table 7.2. Stage N1 is the most misclassified class, which can be also seen by the fact that filters for stage N1 are the least consistent across folds. Moreover, filters that exhibit high activation for stage N1 epochs exhibit high activation for other sleep stages as well. However, there is one stage N1 filter which

appears in more than half of the folds. It has two spikes of activity around 7 Hz and around 9 Hz, and always exhibits activity in high frequencies (20-50 Hz). As noted in Table 6.2, there is evidence in the literature that features from modalities other than EEG, such as eye movements [101], and EMG activity [38, 97] can manifest themselves in the high frequencies of EEG. As shown in Tables 2.2 and 2.3, eye movements and chin EMG tone are features that can differentiate stages N1 and R. We hypothesized that the differences between the stage R and stage N1 filters in the 20-50 Hz frequency range are related to those scoring rules. As in the case of the stage N2 and stage N3 filters described above, extensive prior knowledge and manual tweaking of filters would be required to design those filters for stages N1 and R, while with a CNN these patterns are learned directly from the data.

Finally, there are another two general characteristics in the filters that are consistent across all folds. The first is that there are almost no filters with activity in 10.5-12 Hz. One reason that we believe this is the case is that this is the frequency region in which alpha (8-13 Hz) activity and sleep spindles (12-15 Hz) overlap, which would not be beneficial for distinguishing stages N1 and W from stage N2 (see Table 2.2). The second general characteristic is the absence of 15-17 Hz activity in any of the filters.

A CNN can therefore achieve performance in automatic sleep stage scoring comparable to our state-of-the-art hand-engineered feature approach [94] described in Chapter 6 of this thesis, without utilizing any prior knowledge from the AASM manual [49] that human experts follow, using a single channel of EEG (F_{pz}-C_z). We analyzed and visualized the filters learned by the CNN, and discovered that the CNN learns filters that closely capture the AASM manual's guidelines in terms of their frequency characteristics per sleep stage. Our work shows that end-to-end training in CNNs is not only effective in terms of sleep stage scoring performance, but the CNN model's filters are interpretable in the context of the sleep scoring rules, and are consistent across folds in cross-validation.

Chapter 8

Conclusion

In this thesis we presented and compared two methods for automated sleep stage scoring using single channel EEG, based on deep learning. We achieved state-of-the-art results with the first method, which utilises time-frequency analysis for feature extraction that closely follows the scoring manual that experts follow. We achieved comparable results with the second method, in which we proposed a CNN architecture for automatically learning the filters relevant to the sleep scoring problem without utilising prior knowledge on how those filters should look like.

Outside of automated sleep stage scoring, our work can have applications in other biosignal-based (e.g. EEG and ECG) classification problems. In particular, our analysis and visualisation of the learned filters can prove useful in novel applications for which very little domain knowledge is available. For those applications, analysing and visualising the learned CNN filters can assist in advancing the understanding of the neurophysiological characteristics of a particular condition. Using our methodology CNNs can be turned from an automation tool into a scientific tool.

8.1 Discussion

In terms of the computational burden, the two methods we proposed for sleep scoring are significantly different. On a machine with a processor with 4 cores at 2.8 GHz, 16

GB of RAM and an SSD hard disk, a full run of cross-validation for the stacked sparse autoencoders (SAE) model takes approximately 2 hours, including ensemble learning with 20 models per fold (see Section 6.1). The CNN model took approximately 4 days for a full run of cross-validation (see Section 7.1), which is 48 times longer, only with a single model per fold. On the other hand, at test or estimation time (when the trained model is used) the difference is negligible. Moreover, the speed at estimation time is not a crucial requirement for a sleep scoring system as it is not a real-time system. The immense difference in computational time during training makes it harder to experiment with different hyperparameter settings in the case of the CNN model, and using an ensemble becomes impractical. Considering the impact of ensemble learning in both the unbalanced classes problem and the overall performance of the SAE model, being able to create an ensemble of CNN models could significantly boost performance, and even make the CNN model outperform the SAE model.

A widely used method for EEG signal analysis in the literature is independent component analysis (ICA) (for single channel ICA see [50] and [24]; for ICA in EEG analysis see [75] and [20, pp. 87–9]; for a general treatment of ICA see [41, pp. 560–5] and [39, Ch. 13]). ICA is an unsupervised learning method that aims to separate a signal into independent signals (or sources) that when scaled and added together reproduce the original signal. As has been shown in [24], individual sources can be separated by ICA when those sources are spectrally disjoint, i.e. when their frequency activity does not overlap. In EEG analysis, ICA is commonly used for eye and muscle artifact detection and removal. In sleep scoring, ICA could be used as an alternative or complementary feature engineering method by extracting the amount of activity across the different independent components over time. It is not clear if the criterion of independence, and by extension, the spectral disjointness requirement, would be useful for feature extraction tailored to sleep scoring, as sleep stages frequently exhibit overlapping frequency activity. When departing from a direct application of the sleep scoring rules for feature extraction, an end-to-end learning solution that takes into account the sleep stages, such as the CNN

architecture we proposed in Chapter 7, may be more suitable. ICA would be particularly useful though in detecting artifacts from eye movement, muscle tone, or even cardiac pulse [75]. In Chapter 6 we used gamma activity as a proxy for such artifacts. However, using ICA those artifacts could be captured with more precision.

In this thesis we focused on a single-channel EEG implementation for automated sleep scoring. It is, however, possible to extend the methodologies presented in Chapters 6 and 7 to form the basis for a multi-sensor implementation, for example, for using the algorithms in a sleep clinic setting, in which the aim is to automate sleep scoring without the constraint of using minimal hardware. Doing that could potentially improve the performance of the sleep scoring algorithms, by enabling us to better capture theta activity and higher frequency sleep spindles, which are mostly parietal phenomena [32, 51], and alpha activity, which is predominantly an occipital phenomenon [49]. The P_z-O_z electrode, which is available in the Physionet dataset, is a suitable candidate for that extension. Such a multi-sensor extension would be more straightforward with the SAE model in Chapter 6. Extending the CNN model in Chapter 7 would require multiple adjustments. The main question that would need to be investigated would be at which layer and how to combine the signals from the two or more electrodes. A simple approach would be to have two or more separate sets of convolutional layers for each signal, and then combine the filtered signals at the fully connected layers. However, we hypothesise that it would be beneficial to combine the filtered signals at least at the second convolutional layer, where activity from different frequency bands across electrodes can be combined. Overall, the incorporation of multiple electrodes would result in a network of larger size, with a larger number of parameters to optimise, and, therefore, longer training times.

8.2 Future work

Given that the data that we used contained healthy subjects, we tried to ensure that our algorithms and their evaluation would be robust under different sleep pathologies.

However, large-scale evaluation of our algorithms on datasets with subjects suffering from multiple sleep pathologies is needed. We could then test whether algorithms trained using data from healthy subjects can generalise well to subjects with disorders, and whether disorder-specific algorithms would outperform algorithms trained on a mixture of subjects (healthy and non-healthy). Ideally, we would need datasets for which multiple sleep scorers have scored each single epoch. This way we will be able to quantify the effects of inter-rater variability on our results.

In a real-world sleep monitoring system robustness under displacements of the sensors is also important. In home settings, it is very likely that EEG sensors may be misplaced, or be displaced during the night. The dataset that we used did not contain EEG sensors at adjacent locations, so it was not possible to assess the robustness under displacements. If, for example, a dataset contained electrodes at locations F_z, F_3, F_4, F_{p1} and F_{p2}, in addition to F_{pz}, it would be possible to train our algorithm using the Fpz electrode and test with a 'contaminated' dataset in which a proportion or all of the signals are replaced by one of the F_z, F_3, F_4, F_{p1} or F_{p2} electrodes. To the best of our knowledge, no such study has been done in the literature.

Sleep stage scoring is a significant but by no means the only part of a sleep monitoring system. A real-world sleep monitoring system must account for other aspects of sleep. One example would be movement-related sleep disorders, for which actigraphy could be combined with EEG sensors to get a better understanding of the disorders. We also think that small additions to an EEG-based wearable sleep monitoring system could have significant impact on its practical usefulness. For example, knowing when the lights are out and when they were turned on is a fundamental piece of information that is necessary to record. Adding a simple light sensor to an EEG device could easily provide this information.

We believe that as we move from healthy subjects to sleep pathologies deep learning, and convolutional neural networks in particular, will become even more useful, especially for building disorder-specific models and understanding the particular aspects of sleep

disorders. Such work could lead into the discovery of sleep-related biomarkers for disorders. Moreover, CNNs would be very useful for incorporating modalities other than EEG, such as ECG, for which there is little domain knowledge.

We hope that our work might open up the road for using deep learning in biosignal-based applications without viewing neural networks as 'black boxes', and leveraging their powerful capabilities for scientific research by analysing and visualising what they learn.

Bibliography

[1] Ambroise, C. and McLachlan, G. J. 2002. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of The National Academy of Sciences*, 99(10), pp. 6562–6566.

[2] American Academy of Sleep Medicine. 2014. *The International Classification of Sleep Disorders, Third Edition (ICSD-3)*. Darien, IL: American Academy of Sleep Medicine.

[3] American Psychiatric Association. 2013. *Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-5)*. Washington, D.C.: American Psychiatric Association.

[4] Aserinsky, E. and Kleitman, N. 1953. Regularly occurring periods of eye motility, and concomitant phenomena, during sleep. *Science*, 118(3062), pp. 273–274.

[5] Aserinsky, E. and Kleitman, N. 1955. Two types of unipolar motility occurring in sleep. *Journal of Applied Physiology*, 8(1), pp. 1–10.

[6] Benca, R. M., Obermeyer, W. H., Thisted, R. A. and Gillin, J. C. 1992. Sleep and psychiatric disorders: a meta-analysis. *Archives of General Psychiatry*, 49(8), pp. 651–668.

[7] Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), pp. 1–127.

[8] Bengio, Y. 2012. Practical recommendations for gradient-based training of deep archi-

tectures. In: Orr, G. B., and Mller, K.R. eds. *Neural Networks: Tricks of the Trade*. Berlin: Springer, pp. 437–478.

[9] Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H. 2007. Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J. C. and Hoffman, T. eds. *Advances in Neural Information Processing Systems (NIPS) 19*, Vancouver, 4–9 December, 2006. Cambridge, MA: MIT Press, pp. 153–160.

[10] Berger, H. 1929. Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neurosciences*, 87(1), pp. 527–570.

[11] Berthomier, C., Drouot, X., Herman-Stoca, M., Berthomier, P., Prado, J., Bokar-Thire, D., Benoit, O., Mattout, J. and d'Ortho, M. P. 2007. Automatic analysis of single-channel sleep EEG: validation in healthy individuals. *Sleep*, 30(11), pp. 1587–1595.

[12] Blake, H., Gerard, R. and Kleitman N. 1939. Factors including brain potentials during sleep. *Journal of Neurophysiology*, 2, pp. 48–60.

[13] Blake, H., and Gerard, R. W. 1937. Brain potentials during sleep. *American Journal of Physiology*, 119, pp. 692–703.

[14] Buysse, D. J., Ancoli-lsrael, S., Edinger, J. D., Lichstein, K. L. and Morin, C. M. 2006. Recommendations for a standard research assessment of insomnia. *Sleep*, 29(9), pp. 1155–1173.

[15] Casson, A. J., Yates, D. C., Duncan, J. S. and Rodriguez-Villegas, E. Wearable electroencephalography. 2010. *IEEE Engineering in Medicine and Biology Magazine*, 29(3), pp. 44–56.

[16] Caton, R. 1875. The electric currents of the brain. *British Medical Journal*, 2, p. 278.

[17] Cecotti, H., Eckstein, M. P. and Giesbrecht, B. 2014. Single-trial classification of event-related potentials in rapid serial visual presentation tasks using supervised spatial filtering. *IEEE Transactions on Neural Networks and Learning Systems*, 25(11), pp. 2030–2042.

[18] Cho, S. P., Lee, J., Park, H. D. and Lee, K. J. 2006. Detection of arousals in patients with respiratory sleep disorders using a single channel EEG. In: *Proceedings of the 27th Annual International Conference of the Engineering in Medicine and Biology Society*, Shanghai, 17–18 January, 2006. Washington, D.C.: IEEE, pp. 2733–2735.

[19] Ciresan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. 2012. Deep neural networks segment neuronal membranes in electron microscopy images. In: Pereira, F., Burges, C. J. C., Bottou, L. and Weinberger, K. Q. *Advances in Neural Information Processing Systems (NIPS) 25*, Lake Tahoe, NV, 3–6 December, 2012. Red Hook, NY: Curran Associates, Inc., pp. 2843–2851.

[20] Cohen, M. X. 2014. *Analyzing Neural Time Series Data: Theory and Practice*. Cambridge, MA: MIT Press.

[21] Collura, T. F. 1993. History of electroencephalographic instruments and techniques. *Journal of Clinical Neurophysiology*, 10(4), pp. 476–504.

[22] Costa, M., Goldberger, A. L. and Peng, C. K. 2002. Multiscale entropy analysis of complex physiologic time series. *Physical Review Letters*, 89(6), 068102.

[23] DankerHopfe, H., Anderer, P., Zeitlhofer, J., Boeck, M., Dorn, H., Gruber, G., Heller, E., Loretz, E., Moser, D., Parapatics, S. and Saletu, B. 2009. Interrater reliability for sleep scoring according to the Rechtschaffen & Kales and the new AASM standard. *Journal of Sleep Research*, 18(1), pp. 74–84.

[24] Davies, M. E. and James, C. J. 2007. Source separation using single channel ICA. *Signal Processing*, 87(8), pp. 1819–1832.

[25] Davis, H., Davis, P., Loomis, A. L., Harvey, E. N. and Hobart, B. 1938. Human brain potentials during the onset of sleep. *Journal of Neurophysiology*, 1(1), pp. 24–38.

[26] Decker, M. J., Eyal, S., Shinar, Z., Fuxman, Y., Cahan, C., Reeves, W. C. and Baharav A. 2010. Validation of ECG-derived sleep architecture and ventilation in sleep apnea and chronic fatigue syndrome. *Sleep and Breathing*, 14(3), pp. 233–239.

[27] Dement, W. and Kleitman, N. 1957. Cyclic variations in EEG during sleep and their relation to eye movements, body motility, and dreaming. *Electroencephalography and Clinical Neurophysiology*, 9(4), pp. 673–690.

[28] Dieleman, S. and Schrauwen, B. 2014. End-to-end learning for music audio. In: *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, 4–9 May, 2014. Washington, D.C.: IEEE, pp. 6964–6968.

[29] Edinger, J. D., Fins, A. I., Sullivan Jr, R. J., Marsh, G. R., Dailey, D. S., Hope, T. V., Young, M., Shaw, E., Carlson, D. and Vasilas, D. 1997. Sleep in the laboratory and sleep at home: comparisons of older insomniacs and normal sleepers. *Sleep*, 20(12), pp. 119–1126.

[30] Edinger, J. D. and Morin, C. M. 2012. Sleep disorders classification and diagnosis. In: Morin, C. M. and Espie C. A. eds. *The Oxford Handbook of Sleep and Sleep Disorders*. Oxford: Oxford University Press, pp. 361–382.

[31] Felleman, D. J. and Van Essen, D. C. 1991. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1), pp. 1–47.

[32] Finelli, L. A., Borbély, A. A. and Achermann, P. 2001. Functional topography of the human nonREM sleep electroencephalogram. *European Journal of Neuroscience*, 13(12), pp. 2282–2290.

[33] Fonseca, C., Cunha, J. S., Martins, R. E., Ferreira, V. M., De Sa, J. M., Barbosa, M. A. and da Silva, A. M. 2007. A novel dry active electrode for EEG recording. *IEEE Transactions in Biomedical Engineering*, 54(1), pp. 162–165.

[34] Fraiwan, L., Lweesy, K., Khasawneh, N., Wenz, H. and Dickhaus, H. 2012. Automated sleep stage identification system based on timefrequency analysis of a single EEG channel and random forest classifier. *Computer Methods and Programs in Biomedicine*, 108(1), pp. 10–19.

[35] Gadoth, N., Kesler, A., Vainstein, G., Peled, R. and Lavie, P. 2001. Clinical and polysomnographic characteristics of 34 patients with KleineLevin syndrome. *Journal of Sleep Research*, 10(4), pp. 337–341.

[36] Girshick, R., Donahue, J., Darrell, T. and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, 24–27 June, 2014. Washington, D.C.: IEEE , pp. 580–587.

[37] Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K. and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23), pp. 215–220.

[38] Goncharova, I. I., McFarland, D. J., Vaughan, T. M. and Wolpaw, J. R. 2003. EMG contamination of EEG: Spectral and topographical characteristics. *Clinical Neurophysiology*, 114(9), pp. 1580–1593.

[39] Goodfellow, I., Bengio, Y. and Courville, A. 2016. *Deep Learning*. Book in preparation for MIT Press. Draft available at: `http://www.deeplearningbook.org/` [Accessed: 19 January 2016].

[40] Happe, S., Anderer, P., Gruber, G., Klösch, G., Saletu, B. and Zeitlhofer, J. 2002.

Scalp topography of the spontaneous K-complex and of delta-waves in human sleep. *Brain Topography*, 15(1), pp. 43–49.

[41] Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer.

[42] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath T. N. and Kingsbury, B. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6), pp. 82–97.

[43] Hobson, D. E., Lang, A. E., Martin, W. W., Razmy, A., Rivest, J. and Fleming, J., 2002. Excessive daytime sleepiness and sudden-onset sleep in Parkinson disease: A survey by the Canadian Movement Disorders Group. *The Journal of the American Medical Association*, 287(4), pp. 455–463.

[44] Hoshen, Y., Weiss, R. J. and Wilson, K. W. 2015. Speech acoustic modeling from raw multichannel waveforms. In: *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, 19–24 April, 2015. Washington, D.C.: IEEE, pp. 4624–4628.

[45] Huang, J.T., Li, J. and Gong, Y., 2015. An analysis of convolutional neural networks for speech recognition. In: *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, 19–24 April, 2015. Washington, D.C.: IEEE, pp. 4989–4993.

[46] Hubel, D. H., and Wiesel, T. N. 1959. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3), pp. 574–591.

[47] Hubel, D. H., and Wiesel, T. N. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1), 106.

[48] Hyndman, R.J. and Athanasopoulos, G. 2014. *Forecasting: Principles and Practice*. OTexts.

[49] Iber, C., Ancoli-Israel, S., Chesson, A. and Quan, S. F. 2007 *The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications*. Westchester, IL: American Academy of Sleep Medicine.

[50] James, C. J. and Lowe, D. 2001. Single channel analysis of electromagnetic brain signals through ICA in a dynamical systems framework. In: *Proceedings of the 23rd International Conference of the Engineering in Medicine and Biology Society*, Istanbul, 25–28 October, 2001. Washington, D.C.: IEEE, pp. 1974–1977.

[51] Jobert, M., Poiseau, E., Jähnig, P., Schulz, H. and Kubicki, S. 1992. Topographical analysis of sleep spindle activity. *Neuropsychobiology*. 26(4), pp. 210–217.

[52] Jones, B. E. 2005. From waking to sleeping: Neuronal and chemical substrates. *Trends in Pharmacological Sciences*, 26(11), pp. 578–586.

[53] Kaggle Team. 2015. *Grasp-and-Lift EEG Detection Winners' Interview: 3rd place, Team HEDJ* [Online]. Available at: `http://blog.kaggle.com/2015/10/05/grasp-and-lift-eeg-detection-winners-interview-3rd-place-team-hedj/` [Accessed: 14 June 2016].

[54] Kemp, B., Zwinderman, A. H., Tuk, B., Kamphuisen, H. A. and Oberyé, J.J. 2000. Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomededical Engineering*, 47(9), pp. 1185–1194.

[55] Kiranyaz, S., Ince, T. and Gabbouj, M. 2015. Real-Time Patient-Specific ECG Classification by 1D Convolutional Neural Networks, *IEEE Transactions on Biomedical Engineering*, 63(3), pp. 664–675.

[56] Krizhevsky, A., Sutskever, I. and Hinton, G. 2012. ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C. J. C., Bottou, L. and

Weinberger, K. Q. *Advances in Neural Information Processing Systems (NIPS) 25*, Lake Tahoe, NV, 3–6 December, 2012. Red Hook, NY: Curran Associates, Inc., pp. 1090–1098.

[57] Kushida, C. A., Littner, M. R., Morgenthaler, T., Alessi, C. A., Bailey, D., Coleman Jr, J., Friedman, L., Hirshkowitz, M., Kapen, S., Kramer, M. and Lee-Chiong, T. 2005. Practice parameters for the indications for polysomnography and related procedures: an update for 2005. *Sleep*, 28(4), pp. 499–521.

[58] LeCun, Y. 2015. *In Convolutional Nets, there is no such thing as "fully connected layers"* [Facebook]. Available at: `https://www.facebook.com/yann.lecun/posts/10152820758292143` [Accessed 16 June 2016].

[59] LeCun, Y., Bengio, Y. and Hinton, G. 2015. Deep learning. *Nature*, 521(7553), pp. 436–444.

[60] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), pp. 2278–2324.

[61] Li, F.-F., Karpathy, A. and Johnson, J. 2016. *CS231n: Convolutional Neural Networks for Visual Recognition Course Notes* [Online]. Available at: `http://vision.stanford.edu/teaching/cs231n/` [Accessed: 14 June 2016].

[62] Liang, S. F., Kuo, C. E., Hu, Y. H., Pan, Y. H. and Wang, Y. H. 2012. Automatic stage scoring of single-channel sleep EEG by using multiscale entropy and autoregressive models. *IEEE Transactions on Instrumentation and Measurement*, 61(6), pp. 1649–1657.

[63] Lin, B. C. T., Ko, L. W., Chiou, J. C., Duann, J. R., Huang, R. S., Liang, S. F., Chiu, T. W. and Jung, T. P. 2008. Noninvasive neural prostheses using mobile and wireless EEG. *Proceedings of the IEEE*, 96(7), pp. 1167–1183.

[64] Lockley, S. W., and Foster, R. G. 2012. *Sleep: A Very Short Introduction*. Oxford: Oxford University Press.

[65] Loomis, A. L., Harvey, E. N. and Hobart, G. A. 1937. Cerebral states during sleep, as studied by human brain potentials. *Journal of Experimental Psychology*, 21(2), pp. 127–144.

[66] Martinez, H. P., Bengio, Y. and Yannarakis, G. N. 2013. Learning deep physiological models of affect. *IEEE Computational Intelligence Magazine*, 8(2), pp. 20–33.

[67] Matthews, R., McDonald, N. J., Hervieux, P., Turner, P. J. and Steindorf, M. A. 2007. A wearable physiological sensor suite for unobtrusive monitoring of physiological and cognitive state. In: *Proceedings of the 29th International Conference of the Engineering in Medicine and Biology Society*, Lyon, 23–26 August, 2007. Washington, D.C.: IEEE, pp. 5276–5281.

[68] Mirowski, P., Madhavan, D., LeCun, Y. and Kuzniecky, R. 2009. Classification of patterns of EEG synchronization for seizure prediction. *Clinical Neurophysiology*, 120, pp. 1927–1940.

[69] Mölle, M. and Bergmann, T. O. 2011. Fast and slow spindles during the sleep slow oscillation: disparate coalescence and engagement in memory processing. *Sleep*, 34(10), pp. 1411–1421.

[70] Mourtazaev, M. S., Kemp, B., Zwinderman, A. H. and Kamphuisen, H. A. C. 1995. Age and gender affect different characteristics of slow waves in the sleep EEG. *Sleep*, 18(7), pp. 557–564.

[71] Ng, A., Ngiam, J., Foo, C. Y., Mai, Y., Suen, C., Coates, A., Maas, A., Hannun, A., Huval, B., Wang, T. and Tandon, S. 2015. *Deep Learning Tutorial* [Online]. Available at: `http://deeplearning.stanford.edu/tutorial/` [Accessed: 14 June 2016].

[72] Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Le, Q. V. and Ng, A. Y. 2011. On optimization methods for deep learning. In: Getoor, L. and Scheffer, T. eds. *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellvue, WA, 28 June–2 July, 2011. New York, NY: ACM, pp. 265–272.

[73] National Sleep Foundation. 2016. *Sleep Studies* [Online]. Available at: `https://sleepfoundation.org/sleep-topics/sleep-studies` [Accessed 16 June 2016].

[74] Norman, R. G., Pal, I. Stewart, C., Walsleben, J. A. and Rapoport, R. M. 2000. Interobserver agreement among sleep scorers from different centers in a large dataset. *Sleep*, 23(7), pp. 901–908.

[75] Onton, J. and Makeig, S. 2006. Information-based modeling of event-related brain dynamics. *Progress in Brain Research*, 159, pp. 99–120.

[76] O'Reilly, C., Gosselin, N., Carrier, J. and Nielsen, T. 2014. Montreal Archive of Sleep Studies: An openaccess resource for instrument benchmarking and exploratory research. *Journal of Sleep Research*, 23(6), pp. 628–635.

[77] Palaz, D., Collobert, R. and Doss, M. M. 2013. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *arXiv preprint arXiv:1304.1018*.

[78] Palaz, D., Magimai-Doss, M. and Collobert, R. 2015. Convolutional neural networks-based continuous speech recognition using raw speech signal. In: *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, 19–24 April, 2015. Washington, DC: IEEE, pp. 4295–4299.

[79] PhysioNet. 2002. *The Sleep-EDF database [Expanded]*. Available at: `http://www.physionet.org/physiobank/database/sleep-edfx/` [Accessed 16 June 2016].

[80] PhysioNet. 2002. *The Sleep-EDF database*. Available at: `http://www.physionet.org/physiobank/database/sleep-edfx/` [Accessed 16 June 2016].

[81] Rechtschaffen, A. and A. Kales. 1968. *A Manual of Standardized Terminology, Techniques and Scoring System for Sleep Stages of Human Subjects*. Washington, DC: Public Health Service, U.S. Government Printing Office.

[82] Ren, Y. and Wu, Y. 2014. Convolutional deep belief networks for feature extraction of EEG signal. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. Beijing, 6–11 July, 2014. Washington, DC: IEEE, pp. 2850–2853.

[83] Riemann, D., Berger, M. and Voderholzer, U. 2001 Sleep and depression–results from psychobiological studies: An overview. *Biological Psychology*, 57(1), pp. 67–103.

[84] Rosenblatt, F. 1957. *The Perceptron – A Perceiving and Recognizing Automaton*. Tech. Rep. 85-460-1, Cornell Aeronautical Laboratory.

[85] Ruffini, G., Dunne, S., Fuentemilla, L., Grau, C., Farres, E., Marco-Pallares, J., Watts, P. C. P. and Silva, S. R. P. 2008. First human trials of a dry electrophysiology sensor using a carbon nanotube array interface. *Sensors and Actuators A: Physical*, 144(2), pp. 275–279.

[86] Sainath, T. N., Mohamed, A. R., Kingsbury, B. and Ramabhadran, B. 2013. Deep convolutional neural networks for LVCSR. In: *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, 26–31 May, 2013. Washington, DC: IEEE, pp. 8614–8618.

[87] Silber, M. H., Ancoli-Israel, S., Bonnet, M. H., Chokroverty, S., Grigg-Damberger, M. M., Hirshkowitz, M., Kapen, S., Keenan, S. A., Kryger, M. H., Penzel, T. and Pressman, M. R. 2007. The visual scoring of sleep in adults. *Journal of Clinical Sleep Medicine*, 3(2), pp. 121–131.

[88] Spriggs, W. H. 2014. *Essentials of Polysomnography: A Training Guide and Reference for Sleep Technicians*. 2nd ed. Burlington, MA: Jones & Bartlett Learning.

[89] Stepnowsky, C., Levendowski, D., Popovic, D., Ayappa, I. and Rapoport, D. M. 2013. Scoring accuracy of automated sleep staging from a bipolar electroocular recording compared to manual scoring by multiple raters. *Sleep Medicine*, 14(11), pp. 1199–1207.

[90] Swietojanski, P., Ghoshal, A. and Renals, S. 2014. Convolutional neural networks for distant speech recognition. *Signal Processing Letters*, 21(9), pp. 1120–1124.

[91] Taheri, B. A., Knight, R. T. and Smith, R. L. 1994. A dry electrode for EEG recording. *Electroencephalography and Clinical Neurophysiology*, 90(5), pp. 376–383.

[92] Taigman, Y., Yang, M., Ranzato, M. A. and Wolf, L. 2014. Deepface: Closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, 24–27 June, 2014. Washington, DC: IEEE, pp. 1701–1708.

[93] Thase, M. E., Simons, A. D. and Reynolds, C. F. 1996. Abnormal electroencephalographic sleep profiles in major depression: association with response to cognitive behavior therapy. *Archives of General Psychiatry*, 53(2), pp. 99–108.

[94] Tsinalis, O., Matthews, P. M. and Guo, Y. 2016. Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders. *Annals of Biomedical Engineering*, 44(5), pp. 1587–1597.

[95] Tsinalis, O., Matthews, P. M., Guo, Y. and Zafeiriou, S. 2016. Automatic sleep stage scoring with single-channel EEG using convolutional neural networks. *arXiv preprint arXiv:1610.01683*.

[96] Weisstein, E. W. 2016. *Convolution Theorem* [MathWorld–A Wolfram Web Resource]. Available at: `http://mathworld.wolfram.com/ConvolutionTheorem.html` [Accessed 16 June 2016].

[97] Whitham, E. M., Lewis, T., Pope, K. J., Fitzgibbon, S. P., Clark, C. R., Loveless, S., DeLosAngeles, D., Wallace, A. K., Broberg, M. and Willoughby, J. O. 2008. Think-

ing activates EMG in scalp electrical recordings. *Clinical Neurophysiology*, 119(5), pp. 1166–1175.

[98] Wilson, S. and Nutt, D. 2013. *Sleep Disorders*. 2nd ed. Oxford: Oxford University Press.

[99] World Health Organisation. 2010. *International Classification of Diseases, Tenth Revision (ICD-10)*. Geneva: World Health Organization.

[100] Wulff, K., Gatti, S., Wettstein, J .G. and Foster, R. G. 2010. Sleep and circadian rhythm disruption in psychiatric and neurodegenerative disease. *Nature Reviews Neuroscience*, 11(8), pp. 589–599.

[101] Yuval-Greenberg, S., Tomer, O., Keren, A. S., Nelken, I. and Deouell, L. Y. 2008. Transient induced gamma-band response in EEG as a manifestation of miniature saccades. *Neuron*, 58(3), pp. 429–441.

[102] Zhang, H., McLoughlin, I. and Song, Y. 2015. Robust sound event recognition using convolutional neural networks. In: *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, 19–24 April, 2015. Washington, DC: IEEE, pp. 559–563.

[103] Zhu, X., Zheng, W. L., Lu, B.L., Chen, X., Chen, S. and Wang, C. 2014. EOG-based drowsiness detection using convolutional neural networks. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. Beijing, 6–11 July, 2014. Washington, DC: IEEE, pp. 128–134.