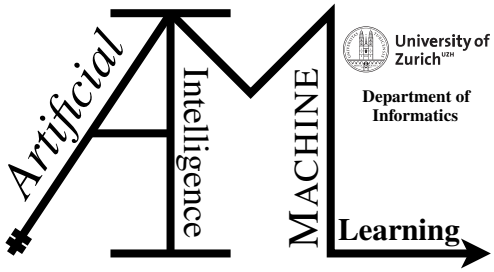
Deep Learning with Temporal Context for Sleep Stage Classification

Master Thesis

Gabriele Brunini
20-742-219

Submitted on
July 16 2023

Thesis Supervisor
Dr. André Anjos
Prof. Dr. Manuel Günther



Master Thesis

Author: Gabriele Brunini, gabriele.brunini@uzh.ch

Project period: January 16 2023 - July 16 2023

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

I extend my heartfelt gratitude to my parents and family for their unwavering emotional and financial support throughout my academic journey. Their encouragement and belief in my aspirations have been instrumental in my accomplishments. I would also like to express my sincere appreciation to my coaches, supervisors, and professors, whose faith in my abilities and directing efforts have played a pivotal role in my growth and achievement. I want to especially express my special gratitude to Prof. Dr. Manuel Günther and Dr. André Anjos for their continuous and qualified support in the thesis research direction and writing process. Finally, I am grateful to my dear friends, whose constant companionship have provided me with emotional support.

Abstract

Detecting and solving sleep disorders can significantly impact society and the economy in general. The polysomnogram is the gold standard exam for diagnosing sleep disorders. Manually annotating the patient's sleep has limitations, including its time-consuming and tedious nature, lack of reliability, sensitivity to the setup of different clinics, and motion noise. This work tests the ability of neural network models to be faster and more reliable than manual scoring by incorporating temporal information in the training setting and changing the model architecture. The study concentrates on algorithms that are robust to the setup of different clinics and fair to diverse populations, using an intelligent combination of the most used datasets in experimental settings: the Sleep-EDF and the MASS datasets. We first analyze the ability of the automated classifier to handle data from different sleep centers and patient groups by experimentally testing loss functions and other crucial model parameters across datasets. Then, we incorporate temporal context in the data samples by concatenating previous sleep epochs to the current sample. We show that our model trained on longer temporal context performs equally to many of the analyzed manual sleep stage scoring conducted by expert technicians and is superior to some state-of-the-art models we analyzed.

Zusammenfassung

Die Erkennung und Behebung von Schlafstörungen kann erhebliche Auswirkungen auf die Gesellschaft und die Wirtschaft im Allgemeinen haben. Das Polysomnogramm gilt als die Standarduntersuchung zur Diagnose von Schlafstörungen. Die manuelle Aufzeichnung des Patientenschlafs hat allerdings ihre Grenzen, darunter die zeitaufwändige und mühsame Natur, die mangelnde Zuverlässigkeit, die Empfindlichkeit gegenüber der Einrichtung verschiedener Kliniken und Bewegungsgeräusche. In dieser Arbeit wird die Fähigkeit von sogenannten "Neural Networks"getestet, schneller und zuverlässiger zu sein als die manuelle Auswertung, indem zeitliche Informationen in die Trainingseinstellung einbezogen werden und die Modellarchitektur geändert wird. Die Studie konzentriert sich auf Algorithmen, die gegenüber dem Aufbau verschiedener Kliniken robust sind und verschiedenen Populationen gerecht werden. Dabei wird eine intelligente Kombination der am häufigsten verwendeten Datensätze in experimentellen Umgebungen verwendet: die Sleep-EDF- und die MASS-Datensätze. Wir analysieren zunächst die Fähigkeit des automatischen Klassifikators, Daten aus verschiedenen Schlafzentren und Patientengruppen zu verarbeiten, indem wir Verlustfunktionen und andere wichtige Modellparameter in verschiedenen Datensätzen experimentell testen. Dann beziehen wir den zeitlichen Kontext in die Datenproben ein, indem wir frühere Schlafepochen mit der aktuellen Probe verknüpfen. Wir zeigen, dass unser Modell, das auf längeren zeitlichen Kontext trainiert wurde, mit vielen der analysierten manuellen Schlafstadienbewertungen, die von Experten durchgeführt wurden, gleichwertig ist und einigen von uns analysierten State-of-the-Art-Modellen überlegen ist.

Acronyms

Contents

Chapter 1

Introduction

> If sleep does not serve an absolutely vital function, then it is the biggest mistake the evolutionary process has ever made.
>
> *Prof. Emeritus Allan Rechtschaffen*

Insomnia is the most common sleep disorder that significantly affects an individual's mental and physical health, quality of life, and productivity. Research results found by Hafner et al. (2023) show how insomnia and other sleep disorders impact society and the economy in general, without distinction among countries. Most sleep disorders fall into four categories: hypersomnia (excessive daytime sleepiness without obvious explanation), insomnia (trouble falling or staying asleep), circadian rhythm (biological clock) disorders, and parasomnias, which are complex behaviors arising from the sleep period (Mahowald and Schenck, 2005). Solving these sleep disorders impacts the individual's quality of life and how they live (Carley and Farabi, 2016). In order to find a solution, we need to analyze and comprehend sleep better: Analyzing sleep status enables us to comprehend sleep conditions, develop strategies for preventing sleep disorders, and safeguard the sleep health of individuals. The starting point of such analysis is to classify the various stages of sleep using the so-called Polysomnogram.

Identifying the sleep stages from overnight PSG recordings plays an essential role in diagnosing and treating sleep disorders, which affect millions of people (Krieger and Lee-Chiong, 2017). Efficiently classifying sleep phases can be done through the Polysomnogram (PSG) exam, the gold standard for diagnosing sleep disorders. This procedure usually follows: The patient sleeps with the PSG recording devices. The exam involves spending one or several nights at a hospital or a sleep center while wearing various sensors that continuously measure various temporal data, such as electroencephalograms, electrocardiograms, electromyograms, oximetry, and respiration rate. An expert then uses the data collected by these sensors to annotate the PSG into different sleep phases (paradoxical summation, light, moderate, and deep sleep), creating a hymnography. Hypnographs help diagnose sleep disorders and involve a plot of the sleep stages recorded over time.

Traditionally, sleep stages are identified based on the visual inspection of the electroencephalogram recordings (EEG) by trained sleep experts. The EEG data comes from electrodes placed on the patient's head, and the position of such electrodes on the brain is indicated by the so-called 10-20 system, or International 10-20 system, for placing the scalp electrodes in the context of the EEG exam. This method was developed to maintain standardized testing methods and to ensure that a subject's study outcomes could be compiled, reproduced, effectively analyzed, and compared using the scientific method (Herwig et al., 2003). The positions are identified in Figure 1.1.

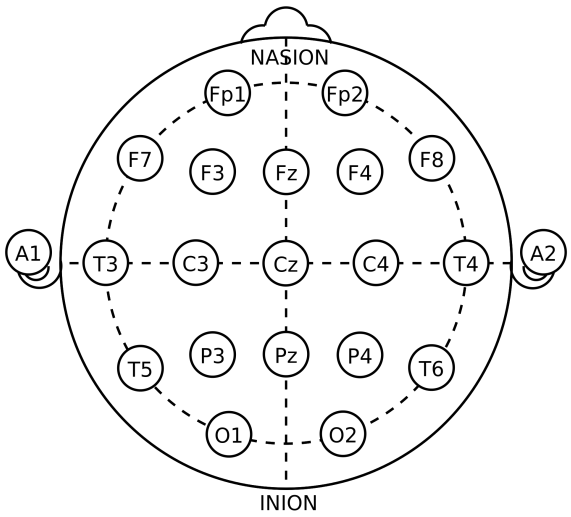


Figure 1.1: THE INTERNATIONAL 10-20 SYSTEM FOR ELECTRODES PLACEMENT. *Image illustrating the placement of the electrodes in the standard configuration, determined by the International 10-20 System (Berry et al., 2017). Each electrode placement site has a letter to identify the lobe or area of the brain it is reading the information; From pre-frontal (Fp), frontal (F), temporal (T), parietal (P), occipital (O), and central (C) areas. The electrode can be placed on the left side of the scalp (odd number), right side (even number), or in the center (Z). Source: Rojas et al. (2018).*

However, manual sleep stage classification is labor-intensive and requires significant expertise, making it prone to subjective biases and errors (Magalang et al., 2013). Each sleep center has its design for data acquisition, and the acquisition process can be affected by involuntary motion during the night, leading to noisy data. For these reasons, researchers have developed automated sleep stage classification systems: These systems use machine learning algorithms to analyze the PSG recordings and automatically classify them into different stages of sleep (Lee et al., 2022). The primary justification is that automatic sleep classification systems make the procedure faster and less prone to subjective biases than manual scoring. In addition to the subjectivity and error associated with manual sleep scoring, there is also the intra- and inter-coder reliability issue to consider. Intra-coder reliability refers to the consistency of scoring by the same scorer, while inter-coder reliability refers to the consistency of scoring between different scorers. Studies have shown that intra- and inter-coder reliability in manual sleep scoring can vary widely, with many of the studies reporting low levels of reliability (Lee et al., 2022). Automated sleep classification can reduce the tediousness, time consumption, and disagreement issues that can arise in the classification task.

Sleep staging usually relies on capturing changes in the spectral properties of the EEG as well as transient events that occur under the different sleep stages. Technically speaking, an overnight PSG recording is usually divided into 30-second epochs, and a stage class is assigned to each epoch. If two or more classes coexist during a single epoch, the assignment goes to the stage comprising the most significant portion of the epoch (Berry et al., 2017). The two main sleep stages are Rapid-eye Movement (REM) sleep and non-REM sleep. During REM sleep, the patient's eyes move rapidly in various directions, and we record a reduced amplitude and faster frequency in the Electroencephalography (EEG) signal. During non-REM sleep, the eye movement stops, brain waves decrease in frequency, and the body prepares to enter deep sleep (Peever and Fuller, 2017). Specifically, there are four phases of non-REM sleep; each stage lasts 5 to 15 minutes. The patient will go through all four phases at night before reaching REM sleep.

Figure 1.2: RAW EOG SIGNAL. *Representation of the EOG signal for blinking and up-gaze motion. Source: Rusydi et al. (2014).*

The American Academy of Sleep Medicine (AASM) and Rechtschaffen and Kales (RK) guidelines are commonly accepted criteria for sleep stage classification. These guidelines provide a standardized framework for the identification and classification of different stages of sleep based on electroencephalography (EEG) patterns. In Sections 1.2 and 1.3, we view the AASM scoring rules more thoroughly.

1.1 EOG Channels

During the recording of the patient's sleep using EEG sensors, blinks and eye movements can significantly cause distortions. The significant potential difference between the retina and the cornea causes the phenomenon. Blinking and moving the eyes cause variations in the electric fields that travel throughout the entire head and can be several times greater than the activity produced by the brain (Gratton, 1998).

The electrical signals produced by the activity of the eyes can be measured using the so-called Electrooculography (EOG) technique, also a non-invasive, portable, easy-to-use, and affordable technology (Bulling et al., 2010). The one-channel EOG montage is designed to pick up horizontal and vertical eye movement from both eyes. EOG signals are stronger in amplitude than EEG ones, so they are easier to detect and more stable across patients (Zhou et al., 2020). In Figure 1.3, we represented the vertical and horizontal configurations for the EOG channel electrodes and the reference electrode on the forehead. In Figure 1.2, we show the EOG signal recorded by the electrodes when the patient blinks and when there is an up-gaze motion in their eyes.

1.2 AASM Rules

The most common standard for recording sleep stages is the AASM. Clinical practice and research widely use these standards. These guidelines define how to classify sleep stages based on EEG, Electromyography (EMG), and EOG signals. The classification of sleep is divided into the following stages:

- During the **W Period**, or wakefulness period, the brain is conscious and responds rapidly to environmental changes. The main types of brainwaves are the alpha waves (α), represented in 1.4a, and beta waves (β), visually reported in 1.4b. The Alpha wave activity accounts for more than 50% of the total brainwave activity, making it a reliable indicator of the awake period. This period is characterized by frequent blinking and rapid eye movements associated with normal or high chin muscle tone. There may also be a brief W period during sleep.

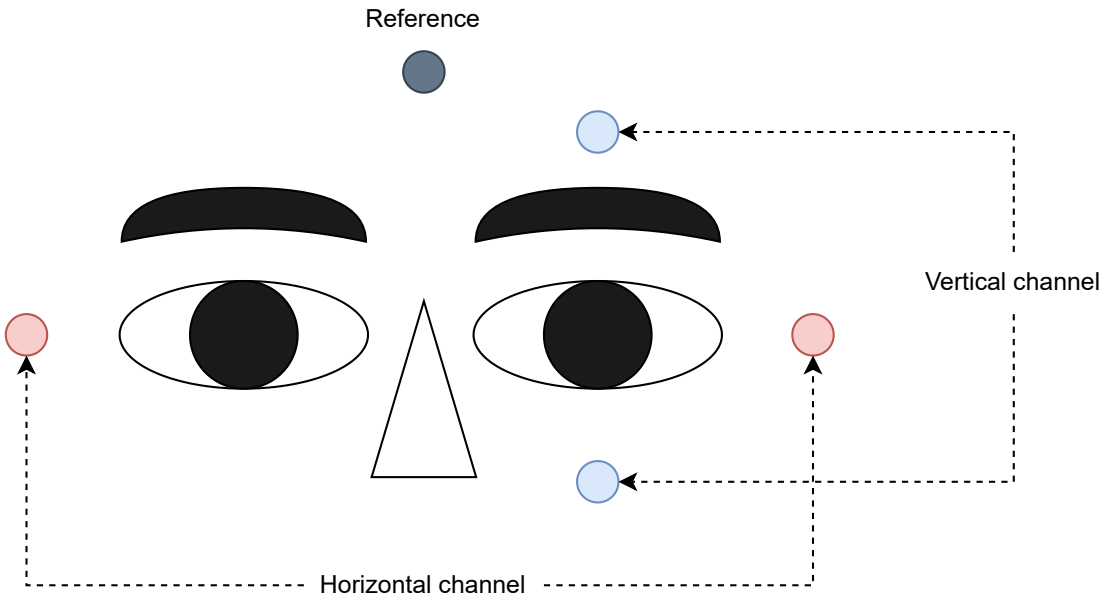


Figure 1.3: EOG SENSORS PLACEMENT. *Representation of the EOG horizontal and vertical channels, represented with two light red electrodes and two light blue electrodes, respectively. The horizontal electrodes are placed on the outer boundary of the eye, while vertical electrodes are placed above and below the eye. The reference electrode is often positioned on the forehead (Belkhiria et al., 2022).*

- The **N1 Stage** is a transitional stage between wakefulness and sleep. During this stage, eye movements slow down and become more sporadic. The sleep state is less susceptible to external stimuli during this period. The proportion of alpha waves in brainwave activity falls below 50%. Low-amplitude theta waves, represented in Subfigure 1.4c, which appear as spikes in the EEG readings predominantly in the 4-7 Hz activity, dominate this stage.
- The **N2 Period** is a deeper stage of sleep compared to N1. During this stage, the amplitude of the EEG signal increases, and eye movements mostly cease. The brainwaves are characterized by spindle and K-complex waves, which are low-amplitude and mixed-frequency. These two structures are visually represented in Subfigure 1.5a and Subfigure 1.5b, respectively. Delta waves account for less than 20% of brainwave activity. Delta waves are represented in Subfigure 1.4d. If the duration between two consecutive spindle or K-complex waves is less than 3 minutes, the experts classify it as N2, while if the duration between the two is longer than 3 minutes, then they classify the stage as N1.
- The **N3 Stage** is a deep sleep state known as slow-wave sleep. During this stage, the brainwaves are dominated by low-frequency delta waves, accounting for over 20% of the total brainwave activity. There are no significant fluctuations in the amplitude of the brainwaves, and eye movements and muscle activity are minimal.
- The **REM Period** is characterized by rapid and autonomous eye movements. The waveform of the EEG during this stage is similar to that of the N1 period, except that the spike-wave is not as apparent. REM sleep is a lighter process during which people remain sensitive to their surroundings.

Table 1.1: COMPARISON BETWEEN THE R&K AND AASM SLEEP STAGING RULES. *The main difference between the two protocols is the merging of N-REM stages 3 and 4 present in protocol R&K into a single stage N3 in the AASM rules.*

AASM	R&K
Stage W	Stage W
N-REM stage 1	N1
N-REM stage 2	N2
N-REM stage 3	**N3**
N-REM stage 4	
REM stage 5	Stage REM

1.3 Rechtschaffen and Kales (R&K) rules

In 1968, Allan Rechtschaffen and Anthony Kales developed the R&K guidelines, and since then, they have been widely used in sleep research (Rechtschaffen, 1968). These guidelines are based primarily on visual scoring of the EEG patterns and include additional criteria such as chin electromyography and eye movements. The R&K guidelines classify sleep into the following stages: The awake state, which indicates when the person is fully conscious and awake (W); the REM sleep, labeled as R; and the Non-rapid Eye Movement (N-REM) sleep which is further divided into four stages, from lightest sleep stage (S1) to the deepest sleep stage (S4), where stages S3 and S4 are referred to as slow-wave sleep (Krakovská and Mezeiová, 2011). The comparison of sleep stage classification between R&K and AASM is shown in Table 1.1. The main difference between the two classification protocols is their total number of classes: While the R&K rules classify sleep into six stages, in the AASM rules, there are only five classes: This is because the N-REM stages 3 and 4 present in R&K are merged into class N3 for AASM protocol.

Over time, new studies and research have led to refinements and modifications in sleep staging criteria. Even if the R&K guidelines are still in use today, it is essential to note that the AASM guidelines have become the standard in recent years and are the primary choice in sleep research and clinical practice. The AASM guidelines provide more detailed criteria for sleep stage classification, incorporating additional features such as sleep spindles and K-complexes. However, some research studies still use the R&K guidelines for historical consistency and comparability with earlier research findings.

1.4 Automatic Feature Extraction and Classification

The use of deep learning for sleep stage classification is nowadays well-known: Deep models can extract features from the data without manual intervention. Some examples of studies involving deep learning for the task have been conducted by LeCun et al. (2015), Tsinalis et al. (2015), Dong et al. (2017), Supratak et al. (2017), and Stephansen et al. (2018). Traditional feed-forward neural networks are unsuitable for time series classification tasks, as they are not specialized in incorporating time information during their training phase, rendering them ineffective for capturing sequential patterns. The Recurrent Neural Network (RNN) architecture is among one of the most viable options. Their architecture works on the principle of storing the output of a layer and feeding it back to the input to predict the output of the layer. However, RNNs present numerous challenges regarding the problem of vanishing and exploding gradients (Ribeiro et al., 2020). This issue arises due to the limited memory capacity during training, causing the network weights to become excessively small or large due to the feedback mechanism.

Convolutional Neural Networks, the cornerstone of deep learning models, have been frequently employed for sleep phase classification (Tsinalis et al., 2016). CNNs are similar to the traditional Artificial Neural Network (ANN) architecture since they comprise neurons that self-optimize through learning. Each neuron will still receive input and operate (such as a scalar product followed by a non-linear function), the basis of countless ANNs. From the input raw vectors of the signal to the final output of the class score, the entirety of the network will still express a single perceptive score function, the weight. The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply (Wu, 2017). However, CNNs use a unique method known as *convolution*. A convolution can be seen as applying and sliding a filter over the time series. Unlike for images, the filters for time series analysis exhibit only one dimension (time) instead of two dimensions (width and height). The filter can also be seen as a generic non-linear transformation of a time series. A general form of applying the convolution for a centered timestep t is given in Equation 1.1.

$$C_t = f\left(\omega * X_{t-l/2:t+l/2} + b\right) \mid \forall t \in [1, T] \tag{1.1}$$

In the equation, C denotes the result of a convolution (dot product $*$) applied on a univariate time series X of length T with a filter ω of length l, a bias parameter b and a final non-linear function f, such as the well-known Rectified Linear Unit (ReLU).

1.5 Research Questions

The following is a general overview of the work we present: We use two different datasets, namely the Montreal Archive of Sleep Studies (MASS)[1] and the Sleep-EDF dataset[2] to train and evaluate different sleep stage classifiers using improved training techniques. The goal is to find a method that works well on both datasets, can be generalized, and outperforms (at best) or draws (at worst) to the human coding benchmark. We explore the following research questions:

- **RQ 1:** Does adding an RNN layer architecture after the Convolutional Neural Network (CNN) feature extractor improve the model's ability to classify the sleep epochs correctly?
- **RQ 2:** Does incorporating temporal context in the data instances improve the sleep stage classification performance and the model's ability to generalize across dataset protocols?
- **RQ 3:** What optimal temporal context length should we use for training the models?
- **RQ 4:** Does using a weighted loss function make a difference when training neural networks across training protocols?
- **RQ 5:** Does adding a horizontal EOG channel to the dataset improve the classification accuracy?

The following sections contain a detailed list of this work's procedure, contributions, and chapters.

1.6 Contributions

The main contributions of this work are the following:

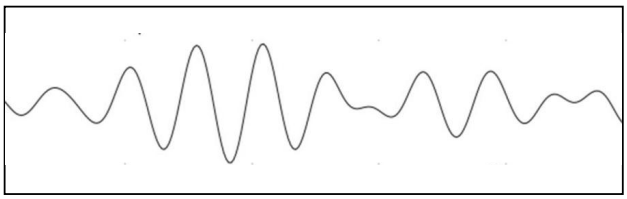
[1] http://ceams-carsm.ca/en/mass/
[2] https://www.physionet.org/content/sleep-edfx/1.0.0/

- We devise a training and testing method on 2-channel EEG data that concatenates multiple sleep epochs together by devising longer sample sequences;
- We modify a state-of-the-art CNN model for sleep phase classification by including an RNN-based layer architecture; The mainly tested layers are the LSTM and GRU architectures.
- We test the best combination of architecture and temporal context-aware training technique on many loss functions, either with class weights adaptation across training epochs or with fixed class weights. This comparison finds whether having adaptive class weights improves the final model scoring system.
- We test our best-performing model architecture on 3-channeled data by including an additional EOG Horizontal channel into our data protocols.
- We finally argue that our pipeline structure, from the preprocessing steps until model evaluation, improves the generalizability of the task of sleep phase classification by comparing the obtained results with the state-of-the-art and manual scoring studies' performance.

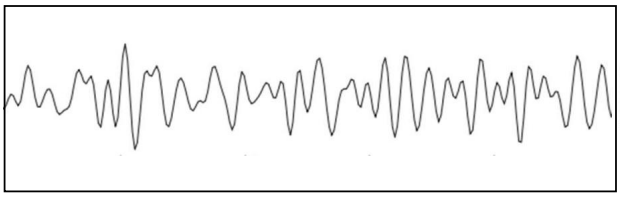
1.7 Thesis Outline

We structured the thesis in the following way:

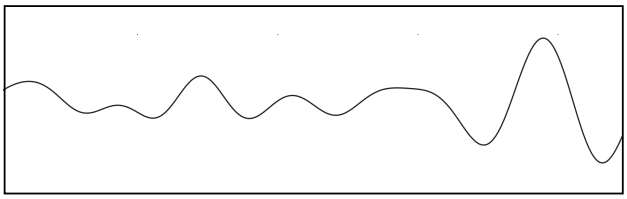
- Chapter 2 introduces common and not-so-common approaches when dealing with sleep phase classification techniques and presents a theoretical analysis of these approaches.
- In Chapter 3, we give a theoretical overview of the state-of-the-art sleep phase classification techniques, introducing the techniques which we will be using for our experiments. We mainly introduce related work concerning CNN and RNN model architectures.
- Chapter 4 gives an overview of the two datasets and data protocols we use to evaluate various training methods and deep learning architectures. Additionally, we define the cross-dataset evaluation procedures we use.
- In Chapter 5, we introduce the novelty of our work in terms of temporal context, training generalization across multiple dataset protocols, and architectural adaptations to CNN model techniques.
- In Chapter 6, we list the experimental trials designed and executed to reach our thesis conclusions.
- In Chapter 7, we share all the results from the experimental setting we defined in the previous chapter. Our analysis is two-fold: First, we look at inter-dataset results to choose the architecture setting that can generalize the best. Then, we take an intra-dataset view of the results and look at the class-specific performance across systems. Finally, we look at the state-of-the-art benchmark and compare the accuracy across systems.
- Chapter 8 compares the intra- and inter-dataset results obtained by our best-performing model to the current manual coding benchmark. For the comparison, we aggregate the results of our model, and we use the Linear Kappa metric. Then, we present the work's final remarks, highlighting the thesis objectives and future developments that could improve the automation and generalization qualities of deep learning techniques.



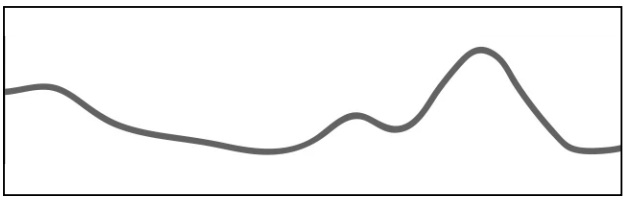
(a) Alpha waves



(b) Beta waves

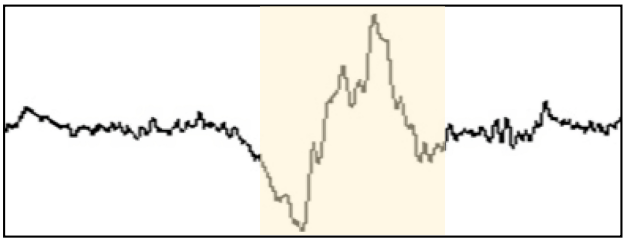


(c) Theta waves

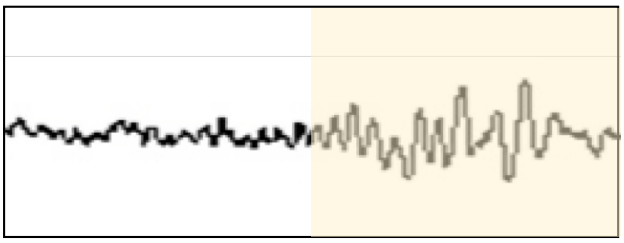


(d) Delta waves

Figure 1.4: BRAIN WAVE TYPES (EEG). *The main types of brain waves are the Alpha waves represented in Subfigure (a) and Beta waves visually reported in Subfigure (b). Subfigure (c) represents Theta waves. Subfigure (d) represents Delta waves, which are present for about 20% of the total brainwave activity.*



(a) Sleep spindle wave



(b) K-complex wave

Figure 1.5: SLEEP SPINDLES AND K-COMPLEX WAVES. *This figure represents sleep splindles in Subfigure (a), and K-complex waves in Subfigure (b). Sleep spindles are found during the early stages of sleep and constitute an electrographic landmark for transitioning from waking to sleep. The K-complex is a waveform seen on EEG during the second stage (N2) of N-REM sleep.*

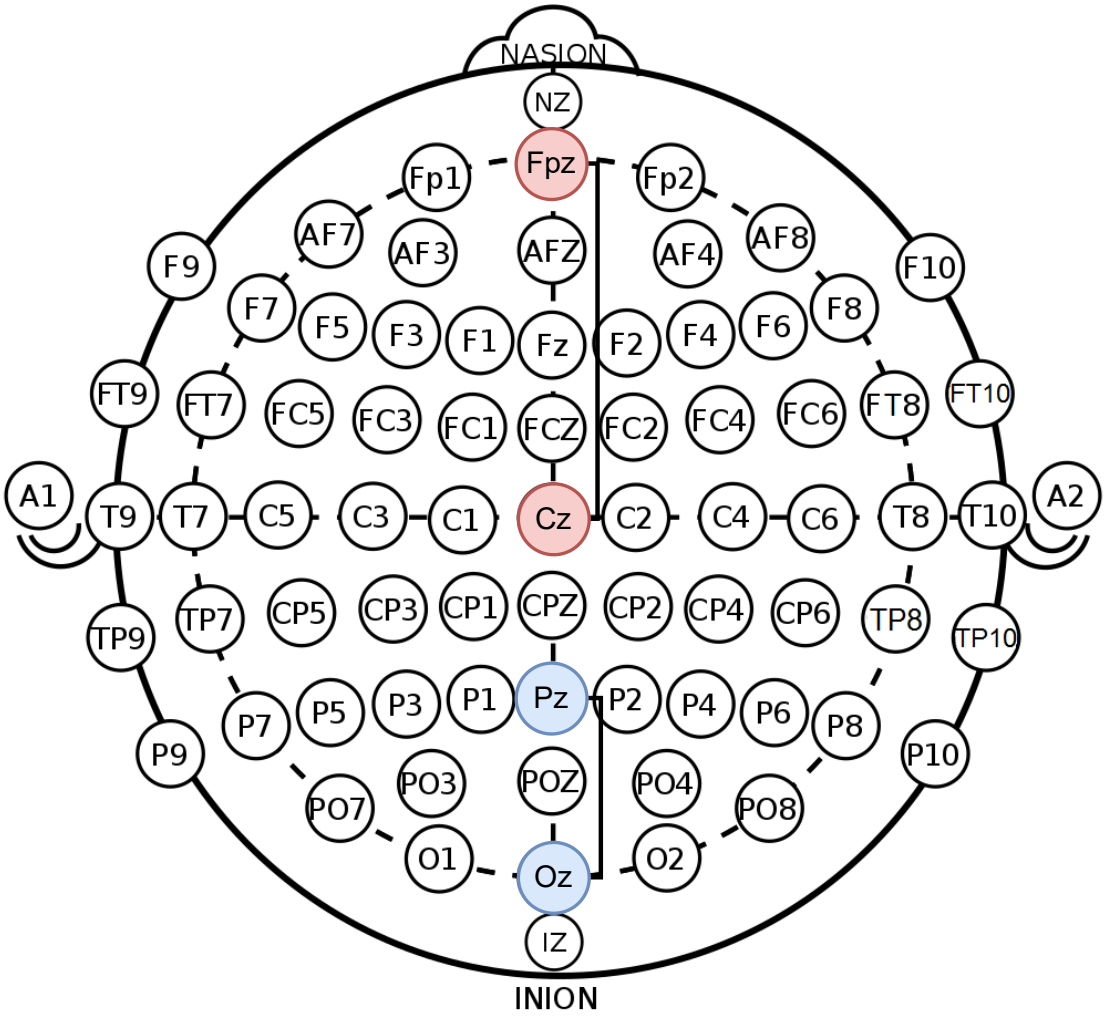


Figure 1.6: THE (EXPANDED) INTERNATIONAL 10-20 SYSTEM. *The extended International 10-20 system position of scalp electrodes. Electrodes are placed at 5%, 10%, and 20% spacings relative to standard skull measurements (e.g., nasion-inion). Abbreviations: A = Auxiliary (Ear lobe, shown, or mastoid, in our experiments), C = central, P = parietal, F = frontal, Fp = frontal polar, O = occipital. In the study, we used two EEG channels connected between electrodes Fpz and Cz (highlighted in red) and between Pz and Oz (highlighted in light blue) for comparability and completeness. Source: Zhang et al. (2020).*

Chapter 2

Related Work

Researchers have investigated automated sleep-scoring methods for years, generating significant research interest. In most reviewed works, the first stage is to preprocess the biological signals to eliminate undesirable information. After preprocessing, biosignal analysis involves one or more feature extraction techniques. Applications for automatic sleep staging use a wide variety of features. Research by Ronzhina et al. (2012) analyzed the latest Artificial Neural Networks (ANN) to effectively and correctly recognize sleep stages. Transformers have become de facto state-of-the-art in Natural Language Processing (NLP) tasks (Vaswani et al., 2017), also proving helpful for other classification tasks, such as sleep stage classification. Analyzing the current trends and techniques, we will review automatic sleep staging methodologies, starting from the filtering procedure as a preprocessing step, the feature extractor techniques, and the state-of-the-art models employed for automatic sleep classification.

2.1 Filters

The procedure of filtering PSG signals occurs to eliminate or reduce the distortion caused by the recording sensors placed on the patient's head. However, filters can also block some electrical activity originating from the brain. Thus, using filters necessitates an equilibrium between filtering out noise and enabling cerebral activity to pass through. These noise reduction techniques substantially affect the EEG signal's final form, feature values, latency, and amplitude. In the early phases of EEG data processing, various filters can be used to remove noise. The most commonly used filters for PSG data are digital filters. They can be high-pass filters, low-pass filters, and similar. The filters are data-dependent, and their frequency highly depends on the studied phenomena.

Digital filters

As analyzed in many EEG studies, the researcher should remove artifacts for correct data interpretation (Karpiel et al., 2021). Digital filtering is the most common technique for preprocessing polysomnographic data and removing such artifacts (Tsinalis et al., 2016). There are always a variety of alterations in the raw EEG signal, including power line disturbances, eye blinking, and muscle movements (electromyographic artifacts). One way to remove the unwanted noise is the notch filter: A signal processing filter that attenuates or removes a specific narrow band of frequencies from a signal. It is typically used to eliminate unwanted noise or interference caused by power line frequencies, such as 50 Hz or 60 Hz, that can contaminate the EEG recordings during sleep studies. In Table 2.1, we reported the relevant studies that show how fragmented the filtering frequency choices are, depending on which channel is processed.

Table 2.1: FREQUENCY FILTERING BY STUDY. *List of relevant studies that describe which channel and which frequency have been filtered. There is no real consensus on the number of filters to apply or the frequency filter to use.*

Study	Filtered channel	Frequency filter
Lan et al. (2015)	EEG	0.5-30 Hz
Zhang et al. (2014)	EEG	0.3-64 Hz
Rodríguez-Sotelo et al. (2014)	EEG	0.5-100 Hz
Imtiaz and Rodriguez-Villegas (2014)	EEG	0.16 Hz + 50 Hz
Wang (2015)	EEG	0.5 Hz + 380 Hz
Popovic et al. (2014); Myllymaa et al. (2016)	EMG	10 Hz + 32 Hz
Zhang et al. (2014)	EMG	0.5-30 Hz
Li et al. (2015)	EMG	12-40 Hz
Liang et al. (2015)	EMG	5-100 Hz
Najdi et al. (2016)	EOG	0.3-35 Hz
Myllymaa et al. (2016)	EOG	0.3-70 Hz
Zhang et al. (2014)	EOG	0.5-30 Hz

The preprocessing of applied filters and other values needs to be structured. The literature exhibits quite a lot of freedom in how the selected methods are used. Each method has its supporters and opponents: Only an expert can diagnose properly and with reliably prepared data. Hence, the influence of the filters on the amplitude and latency calls for a detailed analysis (Karpiel et al., 2021). There needs to be a consensus on filters' number, type, and frequency thresholds that the practicians should use in biological signal processing.

2.2 Hand-crafted Features

Frequently, sleep disorders involve interactions between different sleep stages or abnormal sleep patterns that do not neatly fall into standard classifications. Complex sleep disorders may make it difficult for automated systems to accurately classify sleep stages, necessitating expert interpretation and manual adjustments (Thorpy, 2012). Hand-crafted features are manually conceived and developed using domain-specific knowledge and skill. In the context of sleep stage classification, hand-crafted features identify specific characteristics or patterns in the raw data critical to differentiating sleep stages. These features are meticulously selected and designed to represent particular aspects of the sleep signals, including EEG, EOG, EMG, and other physiological signals (Sun et al., 2019). Features based on frequency and time domains have been used chiefly as input to classifiers (Hasan et al., 2020).

2.3 Trasformer Models

The transformer, a sequence model that relies solely on self-attention, has demonstrated impressive results in various tasks involving sequences, such as natural language processing and sleep stage classification. It consists of an encoder and a decoder, both sharing the same model architecture.

An example of applying the transformer model to classify sleep phases is evident in the research conducted by Phan et al. (2022). In this study, the transformer's multi-head attention module utilizes a scaled dot-product attention mechanism, as illustrated in Subfigure (a) in Figure

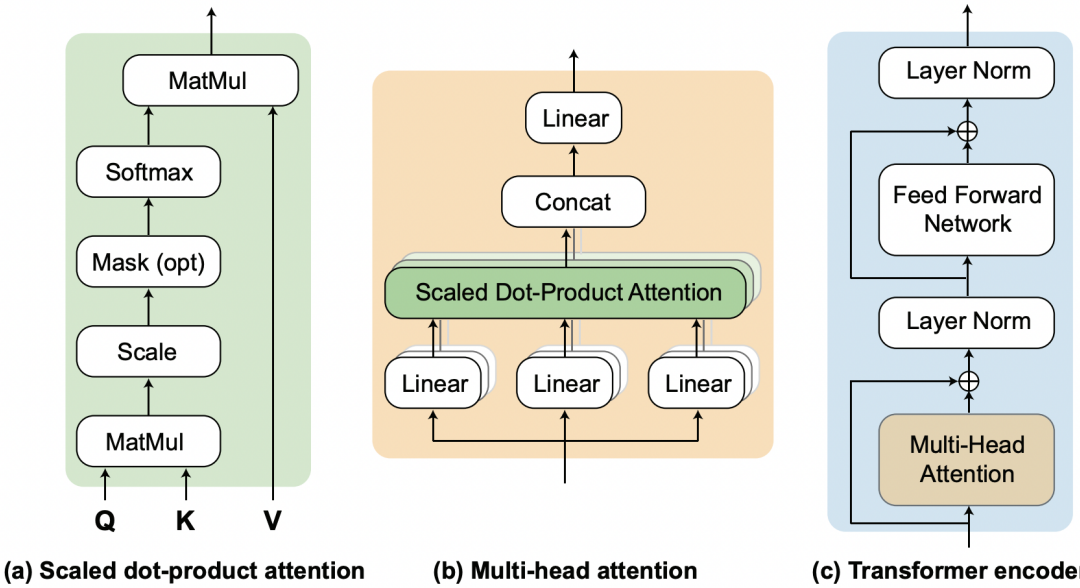


Figure 2.1: TRANSFORMER ARCHITECTURE. *Architecture of (a) scaled dot-product attention, (b) multi-head attention, and (c) transformer encoder. The image is part of the research study by Phan et al. (2022).*

2.1. This mechanism establishes connections between elements at various positions within an input sequence, enabling the generation of the output class. The output is computed by taking a weighted sum of the input values, where the weights for each value are determined through an attention function that involves the query and its corresponding keys.

2.4 Autoencoders

An *autoencoder* is a neural network structure that learns from unlabeled data (Kingma et al., 2019). In this particular neural network architecture, the initial input data undergoes a process known as encoding. The encoder part of the network transforms the input data into a numerical representation, effectively capturing the essential information from the input. This numerical representation, defined as the hidden state, is subsequently passed into a network known as the decoder. The decoder network then generates the output from the hidden state. Figure 2.2 shows a schematic representation of this architecture.

In the context of sleep stage classification, the autoencoders are stacked in an ensemble method and used for classifying sleep stages (Tsinalis et al., 2016). The critical difference between stacked autoencoders and standard neural networks is the layer-wise process of pre-training using unlabelled data (i.e., without class labels) before fine-tuning the network as a whole.

While autoencoders are trained unsupervised, they can also be used in a semi-supervised setting (where part of the data has labels) to improve classification results. In this case, the encoder is used as a feature extractor and is "plugged" into a classification network. Using the encoder as a feature extractor is mainly done in the semi-supervised learning setup, where a large dataset is given for a supervised learning task, but only a tiny portion is labeled.

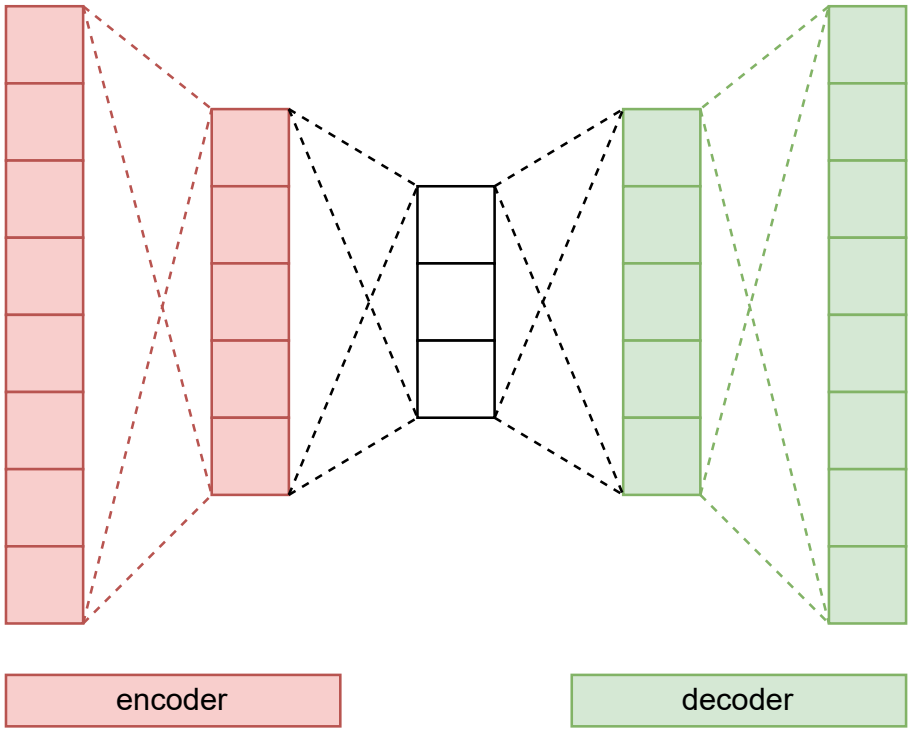


Figure 2.2: AUTOENCODER ARCHITECTURE. *The encoder part, used to compress an input into a lower-dimensional representation, is colored in red, and the decoder part, used to reconstruct the original input from the compressed representation, is colored in green.*

Chapter 3

Background

Much research has been done on sleep stage classification techniques and how to automatize the classification task. Some neural network architectures, such as CNN and RNN models, are ubiquitous in literature, but more advanced and less common techniques exist. This chapter will introduce the most commonly used techniques and a few promising ones, which we will practically demonstrate in the following thesis chapters. Additionally, we will give an overview of the state-of-the-art sleep stage classification, such as preprocessing the time series data, adding temporal context, training using different loss functions, and automatic feature extraction techniques.

3.1 Automatic Sleep Phase Classification

Given the subjectivity of the task, the complexity of the patterns learned by the algorithms, the variability that is present across different patients, and the class imbalances of the epochs to classify across the night, which are not evenly distributed, classifying sleep phases can be a challenging task for a machine learning algorithm (Magalang et al., 2013). Automatic sleep phase classification can enhance the efficiency and accuracy of sleep staging, aid in diagnosing sleep disorders, and contribute to sleep research studies. However, the development and validation of robust and reliable algorithms for sleep stage classification is an ongoing research area, and the performance of these automated systems may vary depending on the quality of the data, the choice of features and algorithms, and the specific characteristics of the population being studied. We explore techniques and model architectures that we use to extract features from raw data automatically.

3.2 Training Neural Networks

The training process for neural network models involves the following steps: We feed the network with input data, typically a batch of training samples. Then, we use the network's output to calculate a loss function, which measures the distance between the predicted output and the ground truth. The network's parameters are updated using the calculated loss, typically with an optimization algorithm like Stochastic Gradient Descent (SGD) or Adam (Kingma and Ba, 2015). This process is repeated for multiple epochs until convergence or some stopping criterion is met (Smith, 2017). The model parameters θ (also defined as weights) are updated at every timestep t as indicated in Equation 3.1, where L is the chosen loss function and $\theta^{(t)}$ is the learning rate at timestep t. It is generally known that choosing a too-small learning rate ϵ will make the training

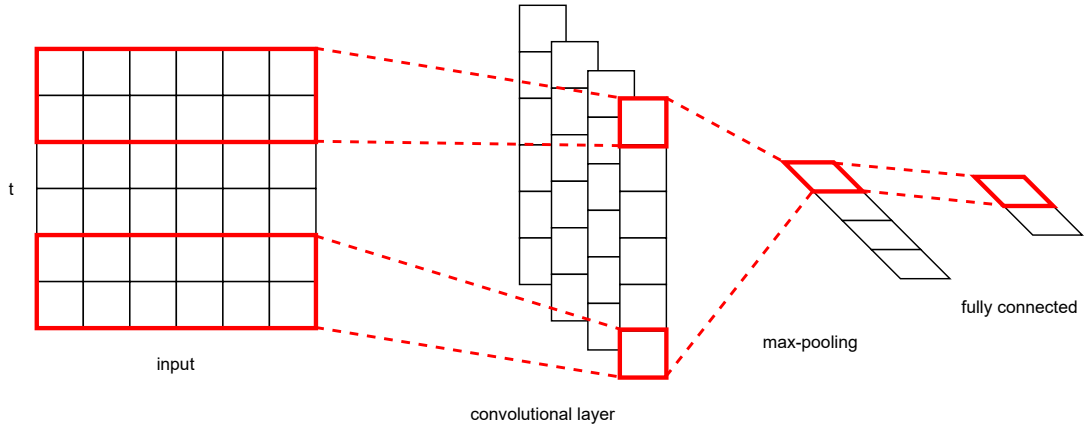


Figure 3.1: CONVOLUTIONAL NEURAL NETWORK REPRESENTATION. *The kernel (squared in red) slides over the input dimensions representing that window. After the convolutional filter goes over all the input regions, the pooling layer calculates the maximum value in each region of the feature map. The most prominent features from the max-pooling layer are then fed into a fully-connected layer.*

algorithm converge slowly while choosing a learning rate that is too large will make it diverge (Bengio, 2012).

$$\theta^{(t)} = \theta^{(t-1)} - \epsilon_t \frac{\partial L}{\partial \theta^{(t-1)}} \tag{3.1}$$

3.3 Convolutional Neural Networks

Convolutional neural networks have been employed in many machine learning tasks, including classifying sleep stages.

Based on the research developments by Fukushima (1980) and LeCun et al. (1998), CNNs have represented a significant step forward in numerous knowledge domains by becoming state-of-the-art for numerous problems. These studies established a layering structure in which each neuron receives a piece of information that is spatially near to each other as input. Each neuron on a layer receives distinct input data from a window that slides across the signal or image. In contrast to traditional neural networks, all neurons in CNNs have identical weights. Consequently, the output of layer l is the convolution of an input feature map $\mathbf{X}^{(l-1)}$ with a set of learnable weights, or filters, $\mathbf{W}^{(l)}$, followed by the addition of a bias term $b^{(l)}$ and the application of a transmission function $f^{(l)}$ to the whole term, as shown in Equation 3.2.

$$\mathbf{X}^{(l)} = f^{(l)}\left(\mathbf{X}^{(l-1)} \cdot \mathbf{W}^{(l)} + b^{(l)}\right) \tag{3.2}$$

Suppose we repeat this schema multiple times in the model architecture. In that case, the result is a network in which each layer extracts features from the previous layer's output only if the spatial relationship of close distance is satisfied (LeCun et al., 2015). Therefore, CNNs typically consist of several convolutional layers that extract the signal or image features. Then a classification technique follows after the CNN structure, such as a fully-connected perceptron or a softmax regression layer, which assigns class probabilities to the outputs based on the extracted features from the convolutional layer. Figure 3.1 contains a visual representation of a standard CNN architecture.

Table 3.1: ARCHITECTURE OF THE MODEL DEVELOPED BY CHAMBON ET AL. (2018). *The detailed architecture of the feature extractor for EEG channels with time series of length T. The identical architecture is utilized for C' EMG channels. The outputs of the dropout layers are concatenated and fed to the final classifier. Since the model is designed for five classes output, the number of parameters for the final dense layer is $5 \cdot ((C + C') \cdot (T//256) \cdot 8)$. Source: Chambon et al. (2018).*

Layer	Layer Type	# Filters	# Parameters	size	stride	Output dimension	Activation	Mode
1	Input					(C, T)		
2	Reshape					$(C, T, 1)$		
3	Convolution 2D	C	$C \cdot C$	$(C, 1)$	$(1, 1)$	$(1, TC)$	Linear	
4	Permute					$(C, T, 1)$		
5	Convolution 2D	8	$8 \cdot 64 + 8$	(1, 64)	$(1, 1)$	$(C, T, 8)$	ReLu	same
6	maxpooling 2D			(1, 16)	$(1, 16)$	$(C, T//16, 8)$		
7	Convolution 2D	8	$8 \cdot 8 \cdot 64 + 8$	(1, 64)	(1, 1)	$(C, T//16, 8)$	ReLu	same
8	maxpooling 2D			(1, 16)	(1, 16)	$(C, T//256, 8)$		
9	Flatten					$(C \cdot (T//256) \cdot 8)$		
10	Dropout (50%)					$(C \cdot (T//256) \cdot 8)$		
11	Dense		$5 \cdot (C + T//256 + 8)$			5	Softmax	

An example of CNNs for sleep stage classification is the work of Chambon et al. (2018). The study's approach is an end-to-end deep convolutional network that contains two blocks: A feature extractor that processes the frequency content of the signal and a fully connected layer that processes a sequence of consecutive 30 seconds-long samples of the signal. The feature extractor processes low-frequency and high-frequency information into two distinct convolutional sub-neural networks before merging the feature representations. The resulting tensor is then fed into a softmax classifier. The softmax function takes vectors of real numbers as inputs and normalizes them into a probability distribution proportional to the exponentials of the input numbers. The softmax function is defined in Equation 3.3, where x_j are the input values.

$$\text{softmax}\,(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \text{ for j} = 1, \ldots, \text{K} \tag{3.3}$$

The model architecture is summarized in Table 3.1. The first layer is the input of dimension (C, T), where C is the number of EEG channels and T is the time series length. The input is reshaped in dimension $(C, T, 1)$ for practical reasons. The layers from 3 to 10 are part of the feature extractor: This part contains three convolutional layers, two max-pooling layers, one permutation, one flattening of the max-pooling output, and a final dropout layer. The dropout probability is set to 50%, meanings half of the neurons will be randomly dropped at every iteration.

3.4 Recurrent Neural Networks

The RNN architecture is usually employed for sequential data modeling tasks, and it is nowadays used for NLP and speech recognition. If the data exhibits sequential nature, where the order of the elements is meaningful, RNN models can come in useful.

Figure 3.2 shows how an RNN works schematically. The main idea behind RNNs is to add a feedback loop inside the model so that the information coming from the previous states of the architecture can be partially reutilized in subsequent states. At time step t, the input to the network is $\mathbf{X}^{(t)}$. Parameter $\mathbf{H}^{(t)}$ represents a hidden state at time t and serves as the "memory" of the network. $\mathbf{H}^{(t)}$ is computed based on the current input and the hidden state of the previous time step. Equation 3.4 is the hidden state at timestep t. We assume the function f is a nonlinear

transformation. All weights (U, V, W) are shared across time.

$$\mathbf{H}^{(t)} = f^{(t)}(\mathbf{U} \cdot \mathbf{X}^{(t)} + \mathbf{W} \cdot \mathbf{H}^{(t-1)}) \tag{3.4}$$

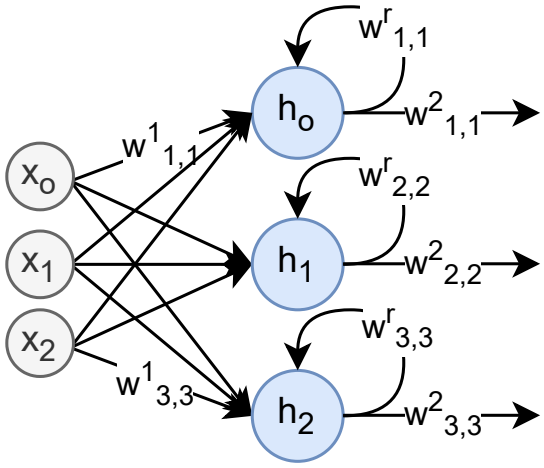


Figure 3.2: RECURRENT NEURAL NETWORK REPRESENTATION. *The blue circles represent the hidden (recurrent) neurons. These recurrent neurons have two connections, one going to the final output and the other going from their output again to their input, and therefore, they have three weights in this example image. This third extra connection is called a feedback loop, and the activation can flow around. In the example, there are 3 data samples (x_0, x_1 and x_2) and 3 hidden units (h_0, h_1 and h_2). The recurrent arrows are labelled as $w^r_{1,1}$, $w^r_{2,2}$ and $w^r_{3,3}$.*

In the research by Phan et al. (2021), a special RNN network called sequence-to-sequence model (often abbreviated to seq2seq) has been applied to solve the sleep phase classification task. As the name suggests, the model takes as input a sequence of items (words, letters, time series) and outputs another sequence of items. In the case of sleep phase classification, the input is a time series of PSG data, and the output is a sequence of features on which classification is made. The most common architecture used to build Seq2Seq models is the Encoder-Decoder architecture.

3.5 LSTM and GRU layers Definition

The problem of vanishing and exploding gradients is typical of RNNs. It arises from the way the partial derivatives of the loss are calculated: They are locally calculated and then combined using the chain rule, which gives a product of many factors. If the individual factors are smaller than 1, their product will be even smaller. If this happens and the gradient becomes very small, i.e., close to 0, then we say the gradient vanishes. To overcome the problem, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers have been designed. The LSTM and GRU layers, defined as gated RNNs, are the solution proposed to regulate the information gathered by the network, from long-term to short-term memory. They learn which information is essential to retain and to discard in the network. The retained info is then used to produce the final predictions.

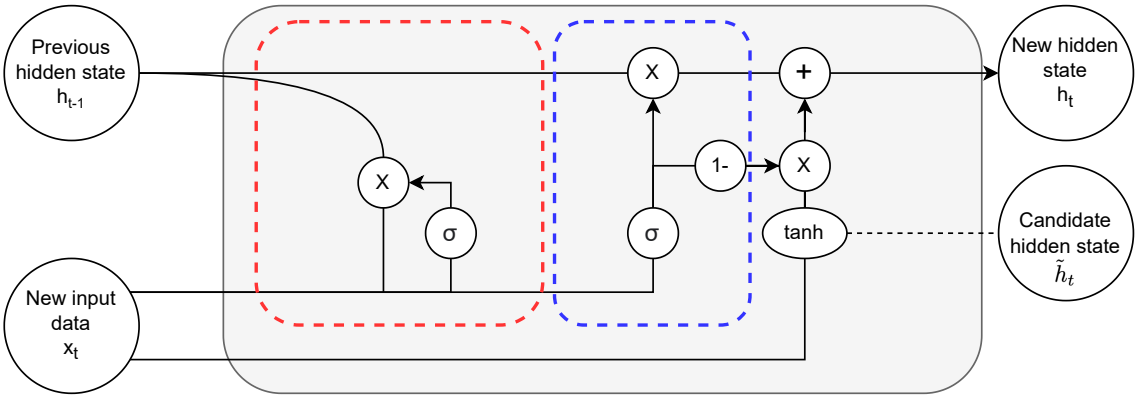


Figure 3.3: GATED RECURRENT UNIT REPRESENTATION. *The reset gate, identified by the red box in the image, decides how much of the past knowledge to forget. The update gate, boxed in blue in the image, determines how much past knowledge can retain as future information.*

3.6 GRU layer

In the field of neural networks, a GRU is an RNN architecture designed to model and process sequential data. Initially introduced by Cho et al. (2014), it is a variation of the more traditional RNNs. It is known for capturing long-term dependencies in sequential data while addressing the vanishing gradient problem. When illustrated, the basic workflow of a GRU is comparable to that of a basic RNN. The primary difference between the two architectures lies in the internal structure of each recurrent unit, as GRU networks are composed of gates that modulate the current input and the previous hidden state. These gating mechanisms include an **update gate** and a **reset gate**. The update gate determines how much input from the previous hidden state is retained, while the reset gate determines how much new input is incorporated into the hidden state. When the new hidden state of the network is calculated, the update gate determines what to collect from the current memory content and what to retain from the previous steps; The two pieces of information are then combined in a weighted sum. The parameters of the GRU are updated using the expressions from Equation 3.5 to 3.8. In the expressions, W_r, W_z, W, U_r, U_z, and U represent the weight matrix of GRU, σ is a logical sigmoid function and z_t represents the update gate, which determines the update degree of the GRU unit's activation value. The r_t parameter is the reset gate, whose update procedure is similar to that of z_t. The $\tilde{h}_t$ parameter represents the candidate hidden layer, and h_t represents the hidden layer.

$$r_t = \sigma\left(W_r \cdot x_t + U_r \cdot h_{t-1}\right) \tag{3.5}$$

$$z_t = \sigma\left(W_z \cdot x_t + U_z \cdot h_{t-1}\right) \tag{3.6}$$

$$\tilde{h}_t = \tanh\left(W \cdot x_t + U \cdot (r_t, h_{t-1})\right) \tag{3.7}$$

$$h_t = [(1 - z_t) \cdot h_{t-1} + z_t, \tilde{h}_t] \tag{3.8}$$

We represent the GRU in Figure 3.3. At each time step, the GRU unit takes the current input data and the previous hidden state as input. It then calculates the activation values for the update and reset gates, which are used to update the current hidden state.

GRUs have been widely used in applications such as natural language processing, speech recognition, machine translation, and time series analysis.

3.7 LSTM layer

An LSTM layer can be highly advantageous in sleep phase classification, primarily due to the sequential nature of sleep data. The ordering and temporal dependencies of sleep stages, including awake, REM (rapid eye movement) sleep, and various non-REM sleep stages, necessitate an approach capable of capturing such dependencies. LSTMs model and capture these temporal relationships, making them an ideal choice for sleep phase classification tasks. LSTM are models with a so-called *forget* gate, which means the model can forget the information that is no longer needed so that the vanishing gradient problem can be solved.

Since LSTM models are capable of sequential modeling, they have been found efficient in capturing long-term sleep stage transition. They are usually utilized to complement other network types, such as CNN (Supratak et al., 2017) and DNNs (Dong et al., 2017). We must note that we solely utilize the hidden unit (the 'state') of the final item in the sequence for making predictions.

The LSTM cells have three different gates: The **input gate**, which indicates whether the cell is updated, the **forget gate**, which controls whether the memory is set to zero, and the **output gate**, that makes the output of the current cell visible. Apart from the gates we have described, we have another element in the LSTM cell: The preceding hidden state vector, which modifies the cell state. It has a tanh activation because it can then distribute the gradients. Hence, it prevents the vanishing/exploding gradient problem because the information can flow longer into the network without causing problems with the gradient calculation. The tanh is defined in Equation 3.9.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.9}$$

We represent an LSTM layer in Figure 3.4. The red dotted box indicates the cell state and represents the long-term memory part of the LSTM layer. It is essential to notice that the weights flowing from the previous state cell and going into the new cell state can only be modified by the multiplication and addition operations in the cell state. The blue dotted box in the graphical representation, defined as forget gate, determines how much of the previous cell state is retained and how much of the long-term memory is remembered. The expression for the forget gate at timestep t is defined in Equation 3.10.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.10}$$

The black dotted box is the input gate, defining the potential long-term memory retained in the network. Then, the sigmoid activation applied to this number defines the percentage of potential memory we want to retain. The sigmoid layer produces a number between 0 and 1, indicating how much of each part can get through. The sigmoid function is defined in Equation 3.11, while the input gate expression at timestep t is formalized in Equation 3.12.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.11}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.12}$$

The cyan dotted box indicates the output gate: How much of the short-term memory we want to retain that should go in the new hidden state, that is, the output of the LSTM unit. The procedure of applying the output gate at timestep t on the previous hidden state is defined in Equation 3.13.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.13}$$

The output of the LSTM h_t is obtained by applying a tanh function on the previous cell state C_{t-1} and multiplying it by the result of the output gate. The output of the LSTM unit h_t is formalized in Equation 3.14.

$$h_t = o_t \cdot tanh(C_{t-1}) \tag{3.14}$$

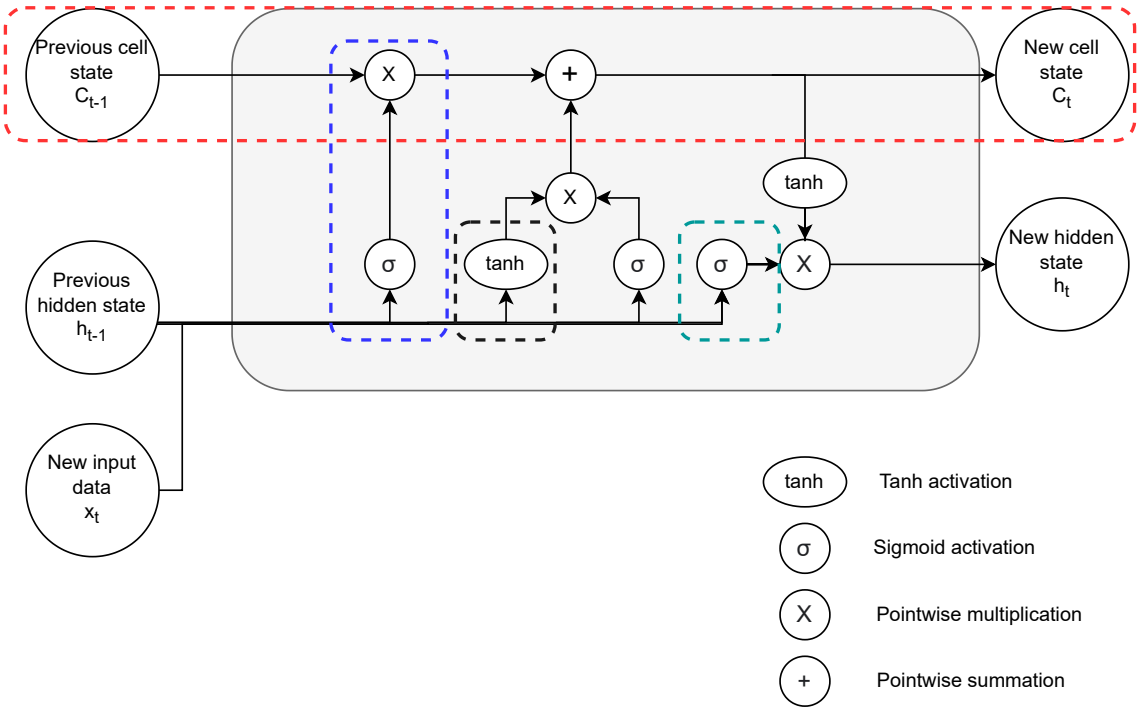


Figure 3.4: LONG SHORT-TERM MEMORY CELL. *The LSTM cell can process data sequentially and keep its hidden state through time. The red dotted box indicates the cell state and represents the long-term memory part of the LSTM layer. The blue dotted box in the graphical representation, defined as forget gate, determines how much of the previous cell state is retained and how much long-term memory is remembered. The black dotted box is the input gate, defining the potential long-term memory retained in the network. The cyan dotted box indicates the output gate: how much short-term memory we want to retain should go in the new hidden state.*

The arrows flowing from the previous hidden state and the new input data represent the short-term memories of the LSTM.

3.8 GRU and LSTM Layers Comparison

The ability of the GRU to hold on to long-term dependencies stems from the computations within the GRU cell to produce the hidden state. While LSTMs have two different states between the cells — the cell and hidden states, which carry the long and short-term memory, respectively — GRUs only have one hidden state transferred between time steps. This hidden state can simultaneously hold both long-term and short-term dependencies due to the gating mechanisms and computations the hidden state and input data undergo. The GRU cell contains only two gates: the Update and Reset gates. Like the ones present in LSTMs, these gates in the GRU are trained to filter out irrelevant information while keeping what is useful selectively. These gates are vectors containing values between 0 to 1, which will be multiplied with the input data and the hidden state. A value of 0 in the gate vectors indicates that the corresponding hidden state is unimportant and will, therefore, return as a zero. On the other hand, a value of 1 in the gate vector means that the corresponding data is essential and will be used.

The main differences between LSTM and GRU lie in their architectures and their trade-offs. LSTM has more gates and parameters than GRU, which gives it more flexibility and expressiveness but also more computational cost and risk of overfitting. GRU has fewer gates and parameters than LSTM, making it simpler, faster, less powerful, and adaptable. LSTM has a separate cell state and output, which allows it to store and output different information, while GRU has a single hidden state that serves both purposes, which may limit its capacity. The two architectures may also have different sensitivities to the hyperparameters, such as the learning rate, the dropout rate, or the sequence length. Compared with the LSTM, the GRU can achieve significantly better results, and its training process is easier, which can improve the training process efficiency (Cho et al., 2014).

3.9 Temporal Context Justification

Sleep is a temporal process with slow stage transitions, implying continuity of sleep stages and strong dependency between consecutive epochs (Iber, 2007; Sousa et al., 2015). Given the strong dependency on consecutive epochs, we can use information coming from the current epoch and the information coming from the previous ones to predict the class of the current epoch. A similar approach has been investigated by Chambon et al. (2018). In the study, the authors propose to aggregate the different features extracted by the CNN model on several time segments preceding *and* following the sample of interest. Then, the obtained vector is fed into the final softmax classifier. This procedure is shown in Figure 3.5.

3.10 Loss Functions

The loss function is a mathematical measure of the model's performance based on its predictions and the actual values. The choice of loss function can significantly impact the performance of the network. In the following subsections, we present four loss functions and their properties.

3.10.1 Cross-entropy Loss

The categorical Cross-entropy (CE) is also known as Softmax Loss. It is a softmax activation plus a Cross-Entropy loss used for multiclass classification. The categorical cross-entropy loss between the true distribution of class j, which we indicate as t_j, and the predicted distribution, indicated as p_j, in a multiclass classification problem with N classes are given by Equation 3.15. In the equation, we multiply the true label t_j with the log of the predicted label p_j. Then, we take the sum over all classes, from 1 to N.

$$\mathrm{CE} = -\sum_{j=1}^{N} t_j \cdot \log\left(p_j\right) \tag{3.15}$$

In a classification problem, the true distribution $[t_1, ..., t_N]$ is a one-hot vector with a value 1 at one of the indices and 0 everywhere else. If a given sleep stage belongs to the class k, in the true distribution vector, t_k = 1, and all other indices are zero. Substituting as $t_k = 1$ if $j = k$ and $t_j = 0$ for $j \neq k$, we see that N - 1 terms in the summation go to zero, and we obtain the simplified expression in equation 3.16, for an input $\in$ class k.

$$\mathrm{CE} = -t_k \log\left(p_k\right) = -\log\left(p_k\right) \tag{3.16}$$

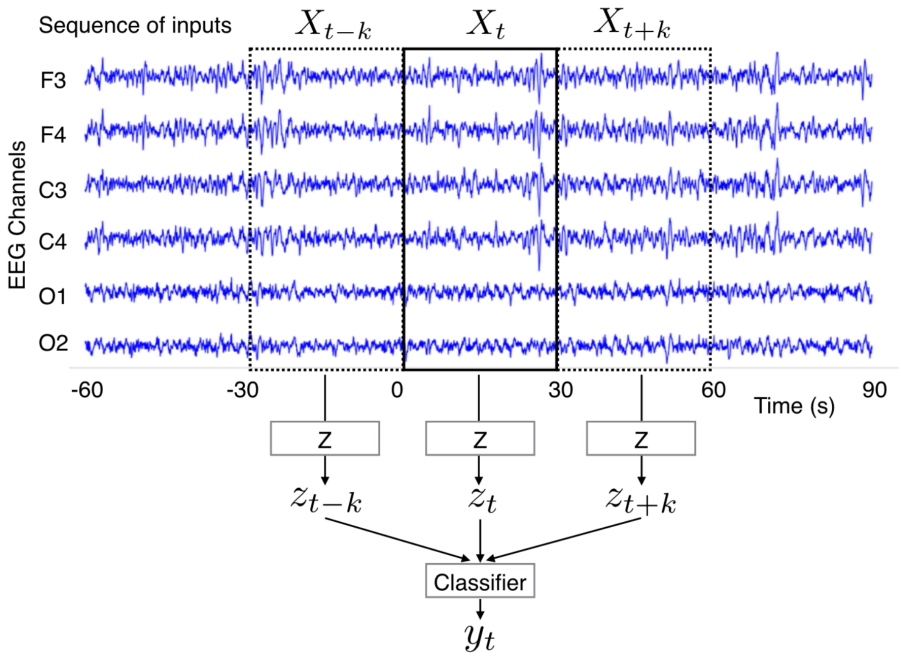


Figure 3.5: TIME-DISTRIBUTED ARCHITECTURE BY CHAMBON ET AL. (2018). *Time distributed architecture to process a sequence of inputs* $\mathcal{S}_t^k = \{X_{t-k}, \cdots, X_t, \cdots, X_{t+k}\}$ *with* $k = 1$. *Element* X_k *represents the multivariate input data over* 30 s *fed into the feature extractor* Z. *Features are extracted from consecutive* 30 s *samples:* $X_{t-k}, \cdots, X_t, \cdots, X_{t+k}$. *Then the obtained features are aggregated. The resulting aggregation of features is finally fed into a dense layer with a softmax classifier to predict the label* y_t *associated with the sample* X_t. *Source: Chambon et al. (2018).*

The loss, therefore, reduces to the negative log of the predicted probability for the correct class. The loss approaches zero, as $p_k \to 1$. The loss is lower when the model predicts a higher probability corresponding to the correct class label. Cross-Entropy aims to take the output probabilities and measure the distance from the actual values.

A standard method for addressing class imbalance in the dataset is to introduce in the loss equation a weighting factor $\alpha \in [0, 1]$ for class 1 and $1 - \alpha$ for class -1. In practice, the term α may be set by inverse class frequency or treated as a hyperparameter to be set by cross-validation. The weighted CE loss for prediction p for class j is defined in Equation 3.17.

$$\mathrm{CE} = -\alpha_{\mathrm{j}} \cdot \log\left(p_{\mathrm{j}}\right) \tag{3.17}$$

3.10.2 Focal Loss

The Focal Loss function addresses class imbalance during training in tasks like object detection. It applies a modulating term to the CE loss to focus learning on hard misclassified examples (Lin et al., 2017). It is a dynamically scaled loss, where the scaling factor decays to zero as confidence in the correct class increases. Intuitively, this scaling factor can automatically down-weight the contribution of easy examples during training and rapidly focus the model on hard examples. The focal loss is defined in Equation 3.18.

$$\mathrm{FL}(p_j) = -\alpha_j (1 - p_j)^{\gamma} \cdot log(p_j) \tag{3.18}$$

This loss function can be interpreted as a binary CE function multiplied by a modulating factor $(1-p_t)^\gamma$, which reduces the contribution of easy-to-classify samples. We note the following properties of the focal loss: When an example is misclassified, and p_t is small, the modulating factor is close to 1, and the loss remains unaffected. As p_t tends to 1, the factor tends to 0, and the loss for well-classified examples is down-weighted. The focusing parameter γ adjusts the rate at which easy examples are down-weighted. The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When the cross-entropy loss functions get overwhelmed because of too much class imbalance in the dataset, an alternative that has been proven to work well is the focal loss function.

3.10.3 MSE Loss

The Mean Squared Error Mean Squared Error (MSE) is perhaps the simplest and most common loss function. The MSE is the sum of the difference between the model's predictions $\hat{y}_i$ and the test values, or ground truth, y_i, squared and averaged across the whole dataset. The MSE loss equation is represented in Equation 3.19, where N is the dataset size.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.19}$$

It is important to note that since we are squaring the errors, the MSE will never be negative. Intuitively, in the continuous case, If the prediction values are the same as the test values, the mean squared error will be equal to zero. The mean squared error increases as the distance between the predictions and the associated test values from the model increase. Thus, a model with a lower mean squared error more accurately predicts the values of the test set.

3.10.4 Kullback-Leibner Divergence

The Kullback-Leibler (KL) divergence is a measure of how a probability distribution differs from another probability distribution. In Bayesian theory, there is some true distribution $P(X)$ that we want to estimate with an approximate distribution $Q(X)$. In this context, the KL divergence measures the distance from the approximate distribution Q to the true distribution P. Mathematically, consider two probability distributions P, Q on some space $\mathcal{X}$. The KL divergence from Q to P (written as $D_{KL}(P\|Q)$) is defined in Equation 3.20. It is important to note that the KL Divergence is not symmetric: that is, $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$. As a result, it is not defined as a distance metric (MacKay, 2003).

$$D_{KL}(P|Q) = \mathbb{E}_{x\sim P}\left[\log\frac{P(X)}{Q(X)}\right] \tag{3.20}$$

The study by Shlens (2007) explains that the KL Divergence can be interpreted as measuring the likelihood that samples represented by the empirical distribution P were generated by a fixed distribution Q. If $D_{KL(p|q)} = 0$, we can guarantee that p is generated by Q. The study also shows that KL Divergence breaks down as the entropy (combining P and Q) minus the entropy of p, defined as $H(p)$. We could use this relationship to define CE as in Equation 3.21.

$$\text{CE} = H(P) + D_{\text{KL}}(P \mid Q) \tag{3.21}$$

Intuitively, the CE is the uncertainty implicit in H(P) plus the likelihood that P could have been generated by Q. If we consider P to be a fixed distribution, CE and $D_{\text{KL}}(P \mid Q)$ differ by a constant factor for all Q. Then, minimizing the CE is equivalent to minimizing $D_{\text{KL}}(P \mid Q)$.

Chapter 4

Data

We introduce the publicly available datasets we have used to run the experiments and answer the research questions of the thesis. The data used for the automatic classifier of sleep stages comes from two sources:

- **Montreal Archive of Sleep Studies**;
- **Sleep-EDF** (also defined as Physionet dataset).

4.1 MASS Dataset

The Montreal Archive of Sleep Studies (MASS) is an open-access database designed to address the methodological challenges faced in sleep research, particularly the time-consuming manual processing of sleep recordings and the lack of a standard benchmark for evaluating automated systems (O'Reilly et al., 2014). The database was pooled from three different sleep laboratories for 200 participants. The recordings include biosignals such as EEG, EOG, EMG, ECG, and respiratory signals.

The MASS database aims to facilitate the development and cross-validation of sleep analysis automation systems by providing a shared resource for the research community. It offers a comprehensive dataset with diverse recordings and a range of signals, enabling researchers to benchmark and validate new analytical tools. Unlike many closed-access databases, MASS promotes open collaboration. It allows for comparing different algorithms using a standard set of benchmarks, which is precisely one of the objectives of our work. By providing an open-access PSG database, MASS aims to improve the reproducibility of research outcomes and promote the identification and improvement of the most reliable systems.

The database includes recordings divided into five subsets, from SS1 to SS5. To have a benchmark to compare our work results, in the work we propose, we decided to use only one of the subsets, SS3, usually used to run the experiments in the literature. This subset contains 62 subjects (age 42.5±18.9 years, with an age range between 20 and 69 years), of which 29 are male subjects (age 40.4±19.4 years, with an age range between 20 and 69 years) and the remaining 33 are female subjects (age 44.2±18.6 years, with age range between 20 and 69 years).

Experienced PSG technicians staged sleep according to either AASM guidelines (Iber, 2007) or the R&K rules (Rechtschaffen, 1968). We report each subset's scoring rules and relevant attributes in Table 4.1.

Table 4.1: MASS ANNOTATION DETAILS. *Details about available annotations for Montreal Archive of Sleep Studies cohort 1 (MASS-C1). The dataset is divided into five night recordings (SS1 to SS5), each with attributes. We have used the highlighted night recording subset SS3 to conduct our experiments.*

	SS1	**SS2**	**SS3**	**SS4**	**SS5**
Sleep stage scoring rules	AASM	RK	**AASM**	RK	RK
Epoch size (in seconds)	30	20	**30**	20	20
Sleep spindles	NA	Expert (in progress)	**NA**	NA	NA
Muscular artefacts	Automatic	NA	**Automatic**	NA	Automatic
Apnea/hypopnea	Expert	NA	**NA**	NA	NA
Micro arousal	Expert	NA	**NA**	NA	NA

4.2 Sleep-EDF Dataset

The second dataset on which we trained and validated all the models is the publicly available Sleep-EDF database from Physionet (Goldberger et al., 2000). The Sleep-EDF database collects 197 whole-night polysomnographic sleep recordings, including EEG, EOG, chin EMG, and event markers. Some records also contain respiration and body temperature data. The corresponding hypnograms, which represent sleep patterns, were manually scored by well-trained technicians according to the Rechtschaffen and Kales (R&K) manual. The database consists of data from two studies: the Sleep Cassette Study (SC-EDF) and the Sleep Telemetry Study (ST-EDF) (Kemp et al., 2000).

The EDF dataset is mainly composed of two components:

- The **ST-EDF** subset comprises PSG recordings from 1987 to 1991. In total, it includes two two-hour recordings for 78 subjects. PSGs of around 9 hours were recorded in the hospital during two nights, one after Temazepam intake and the other after placebo intake. The subjects wore a miniature telemetry system with high-quality signals. The EOG, EMG, and EEG signals were sampled at 100 Hz.

- The **SC-EDF** files (Sleep Cassette Study) involved 153 file recordings obtained from healthy Caucasians aged 25-101 without sleep-related medication. PSGs were recorded for approximately 20 hours during day-night periods at the subjects' homes using a modified cassette tape recorder. The EOG and EEG signals were sampled at 100 Hz, and the submental-EMG signal was sampled at 1 Hz, along with other signals.

All hypnograms were manually scored by well-trained technicians (identified by the eighth letter of the hypnogram filename) according to the manual by Rechtschaffen (1968), but based on Fpz-Cz/Pz-Oz EEGs instead of the commonly used C4-A1/C3-A2 EEG channels, as suggested by Van Sweden et al. (1990).

4.3 EEG Bipolar Reference

EEG is used to measure changes in voltage, which is the difference in electric potential energy between two points in space. In modern EEG, two types of electrode configurations are commonly used: unipolar and bipolar.

In a unipolar configuration, the EEG measures the difference in electric potential between each electrode and a reference electrode. The reference electrodes would ideally pick up all the external noise and interference and no brain-specific fluctuations (Yao et al., 2019). In practical

applications, the reference electrode is typically placed close to the subject's head: This is done to ensure that any environmental interference affects both the reference and measurement electrodes similarly but as far away from the neural sources as possible to prevent the reference signal from picking up brain-based fluctuations.

In a bipolar configuration, two electrodes are put on the head, and the signal generated is the difference between those two electrodes.

EOG and ECG are commonly recorded using bipolar electrodes, and scalp EEG is recorded using unipolar electrodes. The configuration of how the channels are referenced (commonly called a montage) can be easily changed by subtracting pairs of electrodes from each other.

The EEG data from the Physionet dataset was recorded with bipolar reference for channels Fpz-Cz and Pz-Oz. For the MASS dataset, channel Fpz-LER has been computed using the mean of channel Fp1-LER and Fp2-LER, while channel Cz-LER is recorded in the database. Then, we derived two bipolar configurations: channel Fpz-Cz has been computed as the difference between Fpz-LER and Cz-LER, and channel Pz-Oz has been computed as the difference between channels Fpz-LER and Cz-LER. The position of these recorded channels can be seen in Figure 1.6 in the Introduction chapter.

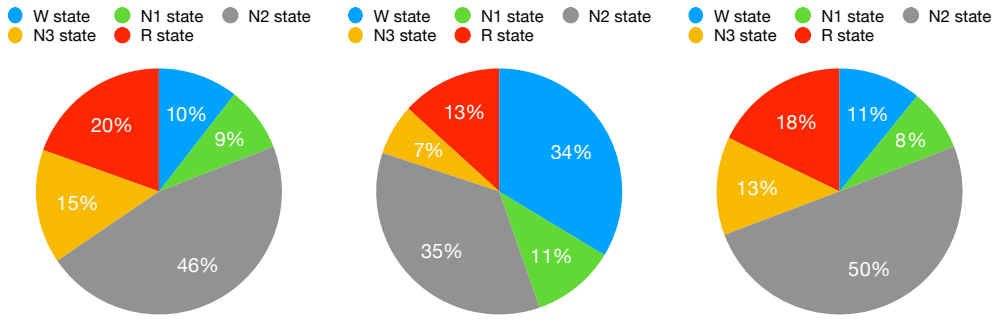
4.4 Resampling

The sampling frequency is different among the Sleep-EDF subsets and the MASS subsets. In Sleep-EDF, the sampling frequency is 100 Hz, while in all subsets of MASS, it is 256 Hz. This detail poses no issue when manually extracting features: However, for extracting features using CNN networks, the filter size must be determined based on the sampling frequency, which needs to be constant across datasets. To fulfill this criterion, we must set a standard sampling frequency across datasets, ensuring we can run our cross-dataset analysis. To encompass the desired frequency range of 0.3 Hz to 30 Hz, we chose a sampling frequency of 100 Hz. The Nyquist theorem states that a sinusoidal function in time can be recovered with no loss of information as long as it is sampled at a frequency greater than or equal to twice per cycle (Colarusso et al., 1999). Therefore, the maximum signal frequency we allow in our time series data is 50 Hz. We resampled the subset SS3 of the MASS dataset to match the 100 Hz frequency and be able to run our cross-dataset analysis.

4.5 Cross-dataset Protocols

The primary objective of this research project is to enhance the generalizability of the algorithms. To achieve this, we have decided to train the algorithms on various combinations of protocols. We test the robustness of the algorithms across different setups by alternatively using samples from different datasets as training, validation, and test sets.

In order to accomplish this, we have constructed datasets with different configurations of setups. We chose subset SS3 from the MASS dataset, divided into epochs of length 30 seconds (instead of 20-second epoch length for subsets SS2, SS4, and SS5) and labeled with the AASM scoring rules. While ST-EDF and SC-EDF share the same channels, only two are shared between MASS (SS3) and EDF (ST and SC). We merged the N-REM sleep stages 3 and 4 into N3 for subsets ST-EDF and SC-EDF (R&K scoring rules) to have the same 5-class configuration in the MASS subset SS3 (AASM scoring rules). A comparison between subsets SC-EDF, ST-EDF, and MASS (SS3) datasets can be seen in Table 4.2. The sleep phase distribution for each subset is visible in Figure 4.1. In the pie charts, it is important to notice the heavy class imbalance across datasets: In



(a) Sleep phase distribution for ST-EDF subset

(b) Sleep phase distribution for SC-EDF subset

(c) Sleep phase distribution for MASS-SS3 subset

Figure 4.1: SLEEP STAGE DISTRIBUTION ACROSS SUBSETS. *Pie charts indicating the percentage of sleep phases per dataset: Subfigure 4.1a for ST-EDF subset, Subfigure 4.1b for SC-EDF subset and Subfigure 4.1c for MASS-SS3 subset. We highlight the underrepresented class N1, which is present only 9% (ST-EDF), 11% (SC-EDF), and 8% (MASS-SS3) of the time.*

subset MASS-SS3, for example, the N1 state accounts for only 8% of the total state distribution, while state N2 accounts for 50%.

To facilitate our experiments and to make them more rigorous, we have designed three protocols, as outlined in Table 4.3. Each protocol contains three subsets: One is used to train and validate the model, while the others are employed for evaluation. We intend to ensure that each dataset can be effectively utilized for training and validating the model and, thus, potentially obtain a more generalized model.

4.6 Intra-dataset Protocol

To run our intra-dataset experiments, we split the subsets into three parts: Training (approximately **60%** of the total epochs), validation (approximately **20%** of the total epochs), and testing (approximately **20%** of the total epochs) splits. Table 4.4 represents the intra-dataset evaluation protocol splits.

Table 4.2: SLEEP-EDF AND MASS-SS3 CHARACTERISTICS COMPARISON. *Description of the two datasets used in this study, Sleep-EDF and MASS, as well as the number of samples available per class for each dataset. The first five rows list the samples available for each sleep class based on the AASM recording standard, while the last five rows describe each dataset attribute.*

	ST-EDF	**SC-EDF-78**	**MASS (SS3)**
W Stage	4488	65'642	6442
N1 Stage	3653	21'520	4839
N2 Stage	19'851	69'132	29'802
N3 Stage	6415	13'039	7653
R Stage	8349	25'835	10'581
Total	**42'756**	**195'168**	**59'317**
# Subjects	22	78	62
# Recordings	44	153	62
Sampling frequency	100 Hz	100 Hz	256 Hz
# EEG channels	2 (Fpz-Cz, Pz-Oz)	2 (Fpz-Cz, Pz-Oz)	20 (BASE, Fp1,Fp2, Fz, Oz)
# EOG channels	1 (Horizontal)	1 (Horizontal)	1 (Horizontal)
Reference	-	-	Linked-ear reference (LER)
Protocol	R&K	R&K	AASM
Age ± Std	40.2 ± 17.7	58.8 ± 22.0	42.5 ± 18.9

Table 4.3: CROSS-DATASET PROTOCOLS. *Table containing the protocol names (EDFxMASS-a, EDFxMASS-d, and EDFxMASS-e) and the allocation of the data subsets (ST-EDF, SC-EDF, and MASS-SS3) in training validation and test sets.*

Protocol	**ST-EDF**	**SC-EDF**	**SS3**
EDFxMASS-a	Train	Test	Test
EDFxMASS-d	Test	Test	Train
EDFxMASS-e	Test	Train	Test

Table 4.4: INTRA-DATASET SPLITTING. *The ST-EDF, SC-EDF, and MASS-SS3 subsets are split into three parts: Training, validation, and testing. We report the number of samples and the number of epochs for each dataset split.*

	train	**validation**	**test**	**Total**
		ST-EDF		
# Samples	28	8	8	44
# Epochs	27'314	7487	7973	42'744
		SC-EDF		
# Samples	97	28	28	153
# Epochs	122'278	38'923	34'559	195'760
		MASS-SS3		
# Samples	38	12	12	44
# Epochs	36'339	11'601	11'377	59'317

Chapter 5

Approach

In this chapter, we lay the groundwork for our experiments by providing a detailed explanation of the procedure and the approach we have used for our work. In the first stage, we train various CNN model variations on cross-dataset and intra-dataset protocols to test their generalization capabilities. The models are initially trained on multivariate datasets using two EEG channels (Fpz-Cz and Pz-Oz). In the second stage, we take a temporal context approach by devising a training method for concatenating multiple sleep epochs. We do this to test whether our model can learn temporal context and to what extent. In the third stage, we implement and evaluate various loss functions that likely impact the final model classification power. In the fourth stage, we add training information by including an additional EOG channel (horizontal configuration) in the datasets and evaluate its impact on the classification task. In the final stage, we compare and discuss the results by considering the state-of-the-art and the inter-rater agreement when manually classifying the sleep phases.

5.1 Signal Pre-processing

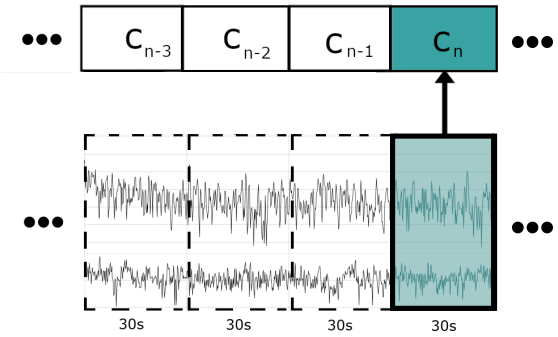
The first important step in learning to read data from PSG and use it to train our model is to pre-process biological signals and remove unwanted content. Such interference can contaminate the extracted features, which may not accurately describe the biological signal studied (Jiang et al., 2019). For our practical purposes, we decided to preprocess all the channels of our signal with a band-pass filter: It is used when we want to transmit signals in a certain band of frequencies and block signals of lower and higher frequencies (Nitschke et al., 1998). Based on medical expert consultation, we have chosen our frequency window to be between 0.3 and 30 Hz. In Section 4.4, we provide additional information on filtering and resampling. The preprocessing was performed using the MNE[1] and Scipy[2] libraries in Python, open-source packages for investigating, visualizing, and analyzing human neurophysiological data.

5.2 Temporal Context Implementation

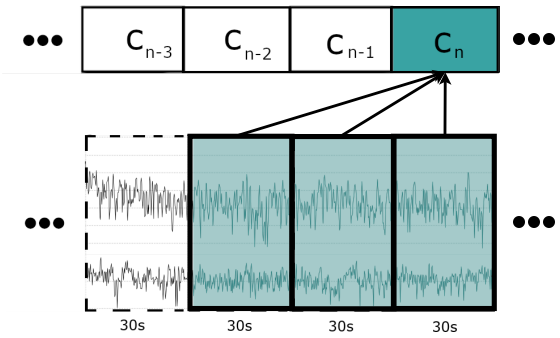
In the temporal context training setting we have utilized for our practical purposes, we concatenate the previous raw samples in the sequence for every sample in our dataset. The model can learn relevant features concerning the current samples and their preceding epochs. Practically speaking, we define parameter *n_past_epochs* that controls how large the considered window

[1] https://mne.tools/stable/index.html
[2] https://scipy.org/

on concatenated past epochs is. This concatenation procedure is visually represented in Figure 5.1, where we compare two example cases: In Subfigure 5.1a, we represented the case where n_past_epochs = 0, that is, no temporal context concatenation occurs in the training samples. In our case, the sample contains 3000 data points (30s length in 100Hz frequency). In Subfigure 5.1b, the case where n_past_epochs = 2 means that the two previous epochs have been concatenated to the current epoch, and the total epoch length becomes 9000 (90s length at 100Hz three times). When we examined the use of temporal context by feeding the predictors with sequences of con-



(a) n_past_epochs = 0



(b) n_past_epochs = 2

Figure 5.1: TEMPORAL CONTEXT REPRESENTATION. *Concatenation of past samples procedure to include temporal context in the training samples. In Subfigure 5.1a, parameter n_past_epochs is set to 0, and the model data is not concatenated. Each class c_n is predicted only using data from the 30s epoch x_n. In Subfigure 5.1b, the n_past_epochs parameter is set to 2, which means that every sample is concatenated with the two previous samples, and the sample length becomes 90 seconds. Class x_n is predicted on 90s-long samples, concatenating epochs x_n, x_{n-1} and x_{n-2}.*

secutive samples, we added zero padding to complete the samples at the beginning of the night, allowing us to feed the models with all the samples of a night record while maintaining fixed

input batch dimensions.

5.3 CNN Baseline Model

The chosen baseline model is based on the model architecture developed by Chambon et al. (2018). The model is divided into two parts: There is a feature extractor part and a classifier part. The Background chapter on Table 3.1 shows the model architecture details. The parameters we have chosen and deemed the most important for the training procedure are the following:

- **batch_size**: This is the number of samples (data points) processed in a single forward/backward pass of the neural network during training. Research shows that in practice when using a larger sample batch, there is a significant decrease in the quality of the model, as measured by its ability to generalize (Keskar et al., 2017). Herefore, we wanted to set a manageable batch size number: We chose the batch size to be 128 samples.
- **criterion**: This is the loss function used to measure the error between the predicted output and the true output of the neural network during training. The goal of training is to minimize this error. We tested different loss criteria for this parameter listed in Section 7.4: weighted CE loss, KL loss, MSE loss, and Focal loss.
- **optimizer**: This is the algorithm used to update the model parameters during training based on the gradients of the loss function. Popular optimization algorithms include SGD and Adam (Kingma and Ba, 2015). We chose Adam as the standard optimization algorithm since empirical results demonstrate that it works well in practice and compares favorably to the other stochastic optimization methods.
- **patience**: The patience value indicates to the trainer how many training epochs must pass after the validation loss has not decreased. If we do not specify a patience value, training will continue for as many training epochs as specified, even if the results are not improving. Setting a patience value saves time and energy by allowing the GPU to rest. In our work, we set the patience value to 10 as default.

5.4 RNN layer Addition to the Baseline CNN model

Adding an RNN layer before the dense layer and softmax classifier used in the study by Chambon et al. (2018) can be justified with study findings in other branches of research. In the study by Gheisari et al. (2021), a combined CNN model (VGG16 and ResNet 50 models) is used to extract spatial features from images for glaucoma detection. The outputs are then passed into a recurrent sequence learning model (i.e., LSTM) to identify temporal features within the image sequence. This method proved successful in their task of glaucoma detection.

In the approach used by Zhang et al. (2018) for relation classification in the NLP research area, a model that combines RNN and CNN in the network structure is proposed to perform a relation classification task. Similarly to our sleep classification task, the traditional classification methods based on the manual feature of lexical resources usually utilize pattern matching and achieve high performance with difficulty. Just as in the manual sleep stage scoring case, the main disadvantage is that manually designing the features is time-consuming and has poor generalization performance because of the low coverage of the training data. Their research proposes a novel model named R-CNN for relation classification. The model first maps the input sentences to low-dimensional vectors. Then they utilize an RNN architecture defined as Bidirectional Long Short-Term Memory Networks (B-LSTM) to capture the context and temporal features of words

Table 5.1: BASELINE CNN MODEL WITH ADDED RNN LAYER. *CNN model architecture developed by Chambon et al. (2018) with added RNN layer. After the CNN feature extractor, we have added an RNN layer to classify the samples into different sleep stages.*

Layer	Layer Type	# Filters	# Parameters	size	stride	Output dimension	Activation	Mode
1	Input					(C, T)		
2	Reshape					(C, T, 1)		
3	Convolution 2D	C	$C * C$	(C, 1)	(1, 1)	(1, T C)	Linear	
4	Permute					$(C, T, 1)$		
5	Convolution 2D	8	$8 * 64 + 8$	(1, 64)	$(1, 1)$	$(C, T, 8)$	ReLu	same
6	max-pooling 2D			(1, 16)	$(1, 16)$	$(C, T//16, 8)$		
7	Convolution 2D	8	$8 * 8 * 64 + 8$	(1, 64)	(1, 1)	$(C, T//16, 8)$	ReLu	same
8	max-pooling 2D			(1, 16)	(1, 16)	$(C, T//256, 8)$		
9	Flatten					$(C * (T//256) * 8)$		
10	Dropout (50%)					$(C * (T//256) * 8)$		
11	**RNN**					$(C * (T//256) * 8)$		
12	Dense					5	Softmax	

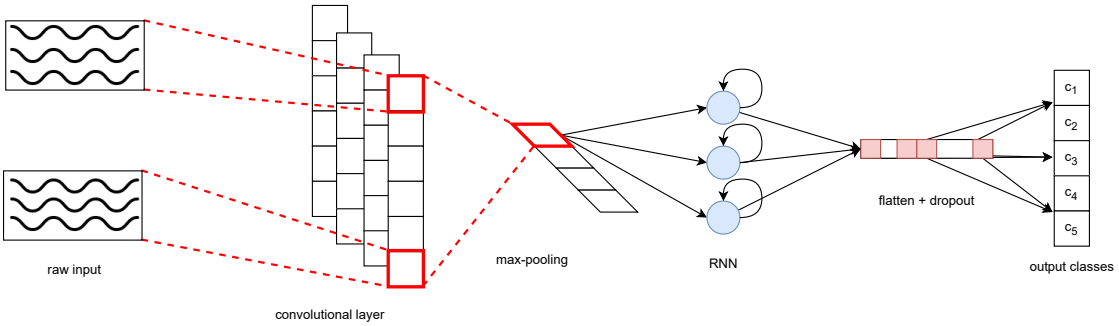


Figure 5.2: COMBINED CNN-RNN ARCHITECTURE. *Architecture of the considered CNN feature extractor, followed by the RNN layer for classification. After the convolutional and max-pooling layers are activated, the extracted features are fed into the RNN classifier. After applying the dropout and flattening of the output, the output is a probability measure for each class. The predicted class is the one with the highest probability assigned.*

in a sentence. We want to follow a similar approach by extracting the input features from the raw epochs using a CNN model architecture and then feeding the output of the CNN to an RNN architecture for classification into sleep stages. This architecture is visually depicted in Figure 5.2.

In our approach, we start from the CNN feature extractor developed by Chambon et al. (2018), and from there, we substitute the final dense layer and softmax activation function with an RNN layer. We tested two RNN systems: the LSTM and the GRU layers, using short-term memory sequential batch modeling. A schematic view of our approach and the model architecture concatenating the CNN feature extractor with the RNN classifier can be seen in Table 5.1. The figure shows that after the data is passed into the convolutional and max-pooling layers, the extracted features are flattened and fed into the RNN classifier of our choice.

Table 5.2: BINARY CONFUSION MATRIX SCHEME. *Confusion matrix in the binary classification case. On the matrix diagonal, the model classifies correctly. The off-diagonal boxes indicate cases where the model predicts the wrong class.*

	Sample is positive	**Sample is negative**
Sample classified as positive	Correct decision: True positive	Type I error: False positive
Sample classified as negative	Type II error: False negative	Correct decision: True negative

5.5 Unbalanced Classification Approach

The machine learning problem we are addressing has the characteristic of being unbalanced: Due to the nature of sleep, the N2 phase is more prevalent than other phases in a healthy person. We have chosen Balanced Accuracy (BA) as our primary evaluation metric. This choice was made because this evaluation approach considers the distribution of classes in the dataset to provide a balanced assessment of the model's accuracy across all classes. Additionally, we report accuracy and Cohen's κ for comparison with existing literature and inter-rater agreement and the simple accuracy score to compare with state-of-the-art approaches.

To define these metrics, we show how the confusion matrix is used to evaluate the model's ability to classify sleep stages correctly. The confusion matrix provides a tabular representation of the model's performance by comparing its predictions with the labels. In the binary classification case, the matrix consists of four critical values: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

- **True Positive (TP)**: The output is a TP if the predicted and true values are positive.
- **True Negative (TN)**: The output is a TN if the predicted and true values are negative.
- **False Positive (FP)**: In this case, the output is an FP if the predicted value is positive but the true value is negative.
- **False Negative (FN)**: Finally, this is the case when the output is an FN if the predicted value is negative, but the true value is positive.

The graphical representation of a confusion matrix in the binary classification case can be seen in Table 5.2. The most commonly used metrics from the confusion matrix are accuracy, precision, recall, and F1 score. Our methodology used accuracy and recall as foundations to evaluate and compare our model variations and experimental setups. We are going to show how they are calculated.

Calculating the accuracy score from the confusion matrix is straightforward: We report the formula in Equation 5.1.

$$\text{Accuracy} = \frac{(\text{TP}+\text{TN})}{(\text{TP}+\text{FP}+\text{FN}+\text{TN})} \tag{5.1}$$

One of the drawbacks of such a metric occurs when dealing with unbalanced datasets, where one class is more common than the others causing the model to classify observations based on this imbalance. This metric does not provide safeguards against a biased classifier that has taken advantage of an imbalanced test set (Brodersen et al., 2010).

Precision measures true positives over the total positives predicted by the model, that is, the rate at which the positive predictions are actually positive. The formula for precision is shown in Equation 5.2.

$$\text{Precision} = \frac{\text{TP}}{(\text{TP}+\text{FP})} \tag{5.2}$$

Recall, also known as sensitivity, measures the true positives over the count of actual positive outcomes. The formula for the recall is expressed in Equation 5.3.

$$\text{Recall} = \frac{\text{TP}}{\text{(TP+FN)}} \tag{5.3}$$

Using this formula, we can assess how well our model can identify the true result.

5.6 Evaluation Metrics

We evaluate the performance of the proposed model architectures by either using the BA score when comparing other model architectures and the Cohen's κ (K) when comparing with the manual stage scoring. We used the accuracy score (ACC) in the intra-dataset analysis to compare our models. The decision on which model performs the best is taken by considering the BA score across datasets, following the inter-dataset analysis.

5.6.1 Balanced Accuracy (BA)

In the context of sleep phase classification with neural networks, the BA score is a metric used to evaluate the performance of a classification model. When dealing with sleep phase classification, the objective is to predict the sleep phase of an individual based on various features or signals. This task typically involves dividing the sleep data into different phases.

In the multi-class classification problem case, the BA is the average recall obtained for each class. When we have N classes, the BA is calculated with the formula in Equation 5.4.

$$\text{BA} = \frac{\sum_{i=1}^{N} \text{Recall}_i}{N} \tag{5.4}$$

In the equation, Recall_i indicates the recall of class i. As discussed in Brodersen et al. (2010), if the classifier performs equally well on either class, this term reduces to the conventional accuracy (number of correct predictions divided by the number of predictions). In contrast, if the conventional accuracy is high only because the classifier takes advantage of an imbalanced test set, the BA, as desired, will drop to chance.

5.6.2 Linear Weighted κ (Cohen's κ)

In the context of sleep phase classification with neural networks, the Linear Weighted κ (also known as Cohen's κ) is a metric used to assess the agreement between predicted and true sleep phases, accounting for the possibility of chance agreement. In our imbalanced classification task, Cohen's κ statistic is a valuable metric: Traditional measures like accuracy, precision, and recall do not offer a comprehensive view of the classifier's performance.

The Linear Weighted κ ranges from -1 to 1, where 1 represents a perfect agreement, 0 indicates agreement equivalent to chance, and negative values indicate disagreement beyond what could be expected by chance.

The observed agreement (o) and the expected agreement by chance (e) are compared to calculate the Linear Weighted κ. The observed agreement represents the proportion of times the model's predictions match the true sleep phases. The expected agreement represents the agreement expected by chance, considering the distribution of sleep phases. The Cohen's κ formula is represented in Equation 5.5.

Table 5.3: COHEN'S κ RESULTS INTERPRETATION. *How to interpret different values for Cohen's κ. Cohen (1960) suggested the κ result be interpreted as follows: values $\leq$ 0 as indicating no agreement and 0.01-0.20 as none to slight agreement, 0.21-0.40 as fair, 0.41-0.60 as moderate, 0.61-0.80 as substantial, and 0.81-1.00 as almost perfect agreement.*

Cohen's κ	**Interpretation**
0	No agreement
0.10-0.20	Slight agreement
0.21-0.4	Fair agreement
0.41-0.60	Moderate agreement
0.61-0.80	Substantial agreement
0.81-0.99	Near perfect agreement
1	Perfect agreement

$$\kappa = \frac{p_o - p_e}{(1 - p_e)} \tag{5.5}$$

In the equation, parameter p_o is the relative observed agreement among raters, and p_e is the theoretical probability of chance agreement. In this sense, the score accounts for the fact that the agreement might have occurred by chance. Cohen's κ compares model predictions with test results when discussing machine learning models. In Table 5.3, we display the interpretation of different values of κ, concerning how much agreement there is in the final classification results.

5.7 Evaluation Protocols

Based on a combination of multiple protocols from EDF and MASS, we evaluate the model framework for experiments on temporal context-based, loss function choice, and multivariate channel choice on a weighted BA metric in an inter-dataset fashion. We weight each BA score on the number of epochs for each dataset combination.

We use the following methodology: Each protocol has a certain number of sample epochs. The number of sample epochs in the testing subsets in the protocol weights the BA obtained for each protocol. Then, to obtain a final score for the model, we aggregate the BA score obtained for each dataset in each protocol and compute a weighted average of all the protocol-specific BA scores. We then use this aggregated score to decide the best model and optimal training setting. Finally, we graphically explore whether our hypothesis holds and whether we have enough evidence to make significant conclusions.

Section 4.5 describes the data protocols we design to test our hypothesis. Each dataset consists of three subsets: train, validation, and test. For each experiment, we reported the BA for each subset. To compare the various models used in our experiments, we need a method to aggregate the results from each subset and obtain a single score. Figure 5.3 illustrates the evaluation protocol we devised for the EDFxMASS protocol. Since we focus on cross-dataset results, we only aggregate the datasets not utilized during training. Specifically, we aggregate the training and validation splits from the subsets which were *not* used for training to compute a validation cross-dataset BA score. The weighted average formula we use is shown in Equation 5.6. The weights used in the calculation are based on the number of epochs in each testing subset. In the equation, parameter w_i is the number of epochs in the i^{th} dataset and the BA is the score achieved by the model in the i^{th} dataset. By applying this weighted average formula, we can effectively aggregate the BA scores across subsets and comprehensively evaluate the models' performance.

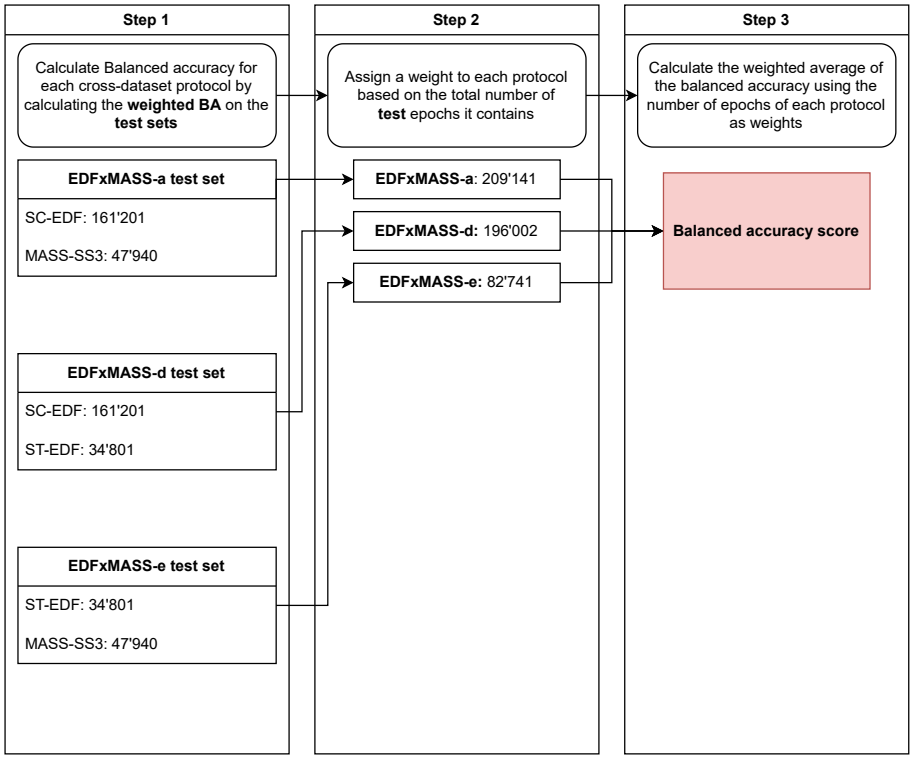


Figure 5.3: INTER-DATASET EVALUATION PROTOCOL. *Three-step approach to evaluate our model architectures in an intra-dataset setting on EDFxMASS protocols. In step 1, we calculate the weighted BA for each test set in the cross-dataset protocol. In step 2, we assign a weight to each protocol based on its epoch size, and in step 3, we calculate the weighted BA result across protocols.*

$$\text{Weighted average} = \frac{\sum_i \text{Balanced Accuracy}_i \cdot w_i}{\sum_i w_i} \tag{5.6}$$

Chapter 6

Experiments

This work investigates different methods for improving sleep stage classification using neural networks, changing their training procedures and essential parameters. This section will overview which experiments we run and which hypotheses we test.

6.1 Experiments

We detail which experiments we have performed to answer the research questions.

In **RQ 1**, we investigate whether adding an RNN layer architecture after the CNN feature extractor improves the model's ability to classify the sleep epochs correctly. The comparison involves the baseline CNN model developed by Chambon et al. (2018), with the same CNN architecture but an RNN layer after the feature extraction part: We tested LSTM and GRU layers as classifiers. Keeping all model and training parameters constant, to perform the intra-dataset analysis, we train all three model architectures on the following subsets: ST-EDF, SC-EDF from EDF dataset, and SS3 subset from the MASS dataset, using the same subset both for training and testing. We use multiple subsets to test our models on data with different statistical properties. Then, we average the balanced accuracy across all subsets by weighting each model accuracy score with the number of epochs contained in the subset. We train on one of the three subsets for the cross-dataset analysis and test our model on the remaining two subsets. Then, we aggregate the test BA results weighting the scores by the number of epochs in each subset. The results of the experiments are presented in Section 7.1.

The objective of **RQ 2** is to investigate whether incorporating temporal context in the data instances improves the sleep stage classification performance and the model's ability to generalize across dataset protocols. We obtain extended sequences with additional temporal context by concatenating previous sample instances with the current training sample. For this experiment, we have created hyperparameter n_prev_epochs to control how many previous samples are concatenated with the current sample fed into the model. We started with the value n_prev_epochs = 0; No temporal context is concatenated, and we tested hyperparameter values up to 7. We repeat this procedure for all the tested models: The baseline CNN model created by Chambon et al. (2018) and the alternatives we have created with additional LSTM and GRU layers. To test whether the model can generalize across protocols, we perform an inter-dataset analysis of the obtained scores by utilizing different subsets for training and testing, averaging the model's balanced accuracy score across the three protocols. The results of the experiments are presented in section 7.2.

In **RQ 3**, we search for the optimal temporal context length for training the models. We, therefore, observe the balanced accuracy score for each model architecture we considered in the

previous experiment as the number of previous epochs concatenated to the current one increases. Then, we aggregate the models' cross-dataset BA results and plot the aggregated result on a line that connects all the accuracy scores. This experiment is run to see the evolution of balanced accuracy while varying parameter n_prev_epochs independently of the tested model architecture. Then, for the intra-dataset analysis, we look at the class-specific BA across sample size for the best-performing model. We report the experimental results in Section 7.3.

For **RQ 4**, we test whether a weighted loss function makes a difference when training neural networks across training protocols. First, we implemented the loss functions in code using the backend of the selected deep learning framework, in our case, Pytorch[1]: A well-known Python framework for machine learning. Then, we select the best-performing model architecture from those tested in the previous experiments. To answer our question, we test four loss functions commonly used for neural networks: weighted CE loss, focal loss, MSE loss, and KL loss. In this experimental setting, we run an experiment after selecting the best-performing model architecture found in the previous experimental steps and the best-performing training sample length across every dataset protocol. Lastly, we average the obtained balanced accuracy scores across dataset protocols, again weighted by the number of epochs in each protocol. The results of this experiment can be found in Section 7.4.

For **RQ 5**, our analysis focuses on adding a horizontal EOG channel to the utilized dataset channels and testing whether the additional information improves the classification accuracy. We design this experiment in the following way: For each model architecture, keeping all parameters fixed, we train and test our models by including the horizontal EOG channel, which is present in both MASS and Sleep-EDF datasets. Then, to have a better overview of our research question, for each protocol in our project setting, we run our experimental analysis on the best-performing model architecture and loss function we have obtained from the previous experimental steps. Keeping the model architecture fixed and adding the horizontal EOG channel described in Section 1.1, we run in a cross-dataset fashion one experiment for each data protocol (EDFxMASS-a, EDFxMASS-d, and EDFxMASS-e) and temporal context length (controlled by parameter n_prev_epochs ranging from 0 up to 10). Afterward, we obtain the cross-dataset balanced accuracy score and aggregate it across datasets, weighting it by the number of sample epochs in each dataset. For the intra-dataset analysis, we compare the results obtained for each tested subset by either including the EOG channel in the subset or not.

[1] https://pytorch.org/

Chapter 7

Results

In this chapter, we go into greater depth on the outcomes of our studies. We employed graphic charts and tables to compare the classifications of the different approaches and model architectures we have utilized and to explain better why we obtained such results. We will present our cross-dataset and intra-dataset results. The appendix chapter includes tables with the results of the experiments run for each dataset. We provide the answers to each research question separately.

7.1 RQ 1: Adding a GRU Layer Improves the classification Performance

We compare the baseline CNN model developed by Chambon et al. (2018) with our modified CNN architectures featuring LSTM and GRU layers. We want to determine whether adding a recurrent layer to the neural network baseline improves the model's ability to classify sleep epochs accurately. We visually represent the answer to this research question in Figure 7.1.

The results indicate that adding an RNN layer to the CNN feature extractor in the baseline CNN model affects the final model results. Adding a GRU layer positively affects the balanced accuracy result: Based on our experiments, the CNN-GRU model achieves 3.39 percentage points more than the baseline CNN model. If we add an LSTM layer instead, this effect is the opposite: We get a balanced accuracy score of 3.83 percentage points *lower* than the one obtained by the baseline model. In our experiment, the CNN-GRU model performs better than the compared architectures.

7.2 RQ 2: Incorporating Temporal Context Improves Classification Performance

We investigate whether adding temporal context improves the sleep stage classification performance in the data instances. We concatenate the sample epochs with previous sample instances. We obtain a more extended time sequence that gives more temporal context to the samples. We run the experiment across datasets using the previously defined EDFxMASS protocols and separately on the three subsets from Sleep-EDF and MASS datasets for the intra-dataset analysis.

The experimental results from Figure 7.2 and Figure 7.3, representing the cross-dataset and intra-dataset results, respectively, show that longer sequences improve the model accuracy. As for the previous experiment, the CNN-GRU performs best among the tested architectures. Looking

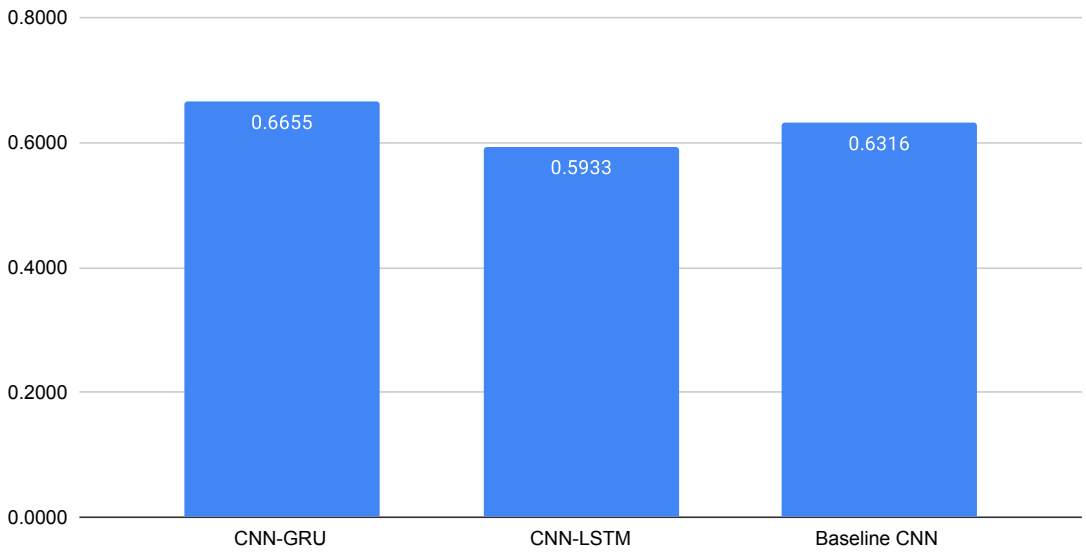


Figure 7.1: RQ 1 - CNN ARCHITECTURES COMPARISON. *Cross-dataset comparison between the baseline CNN model developed by Chambon et al. (2018), and the modified CNN models with added LSTM and GRU units. This experiment does not concatenate sample epochs (n_past_epochs = 0). The graph shows that the CNN-GRU model performs better in this control setting over the other two configurations.*

at the cross-dataset results, the optimal number of past epochs concatenated in each sample, after which accuracy starts to a decade, is 6. The best cross-dataset BA score obtained by CNN-GRU is 0.7220.

Figure 7.4 shows an additional graph analyzing the results. We create the graph by aggregating each tested model cross-dataset results in two bar plots: The blue bar in the plot indicates the case where the training samples have been concatenated to encompass temporal context, that is, to use the concatenated samples in the sleep recording. The blue bar aggregates all the accuracy scores obtained by increasing the model's temporal context from n_past_epochs = 0 to n_past_epochs = 7. The red bar in the plot indicates the case where the sample epochs do not contain additional temporal context (n_past_epochs = 0). The results show that longer sequences improve model accuracy. This result is particularly evident for architecture CNN-GRU, whose average improvement is 3.8 percentage points, compared to an improvement of 1.75 for the baseline CNN model. For the CNN-LSTM architecture, the improvement is even less tangible: By adding temporal context to our samples, we gain only 0.63 percentage points in BA.

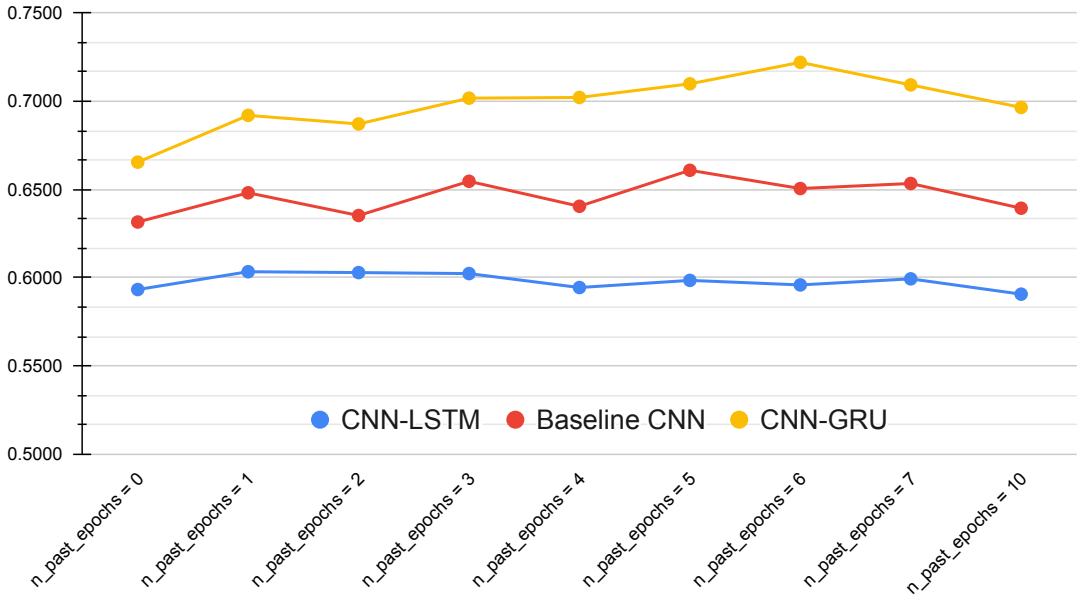


Figure 7.2: RQ 2 - BA CROSS-DATASET RESULTS. *Balanced accuracy results across epochs. The graph shows the aggregated BA for each tested model architecture as the parameter n_previous_epochs ranges from 0 to 10.*

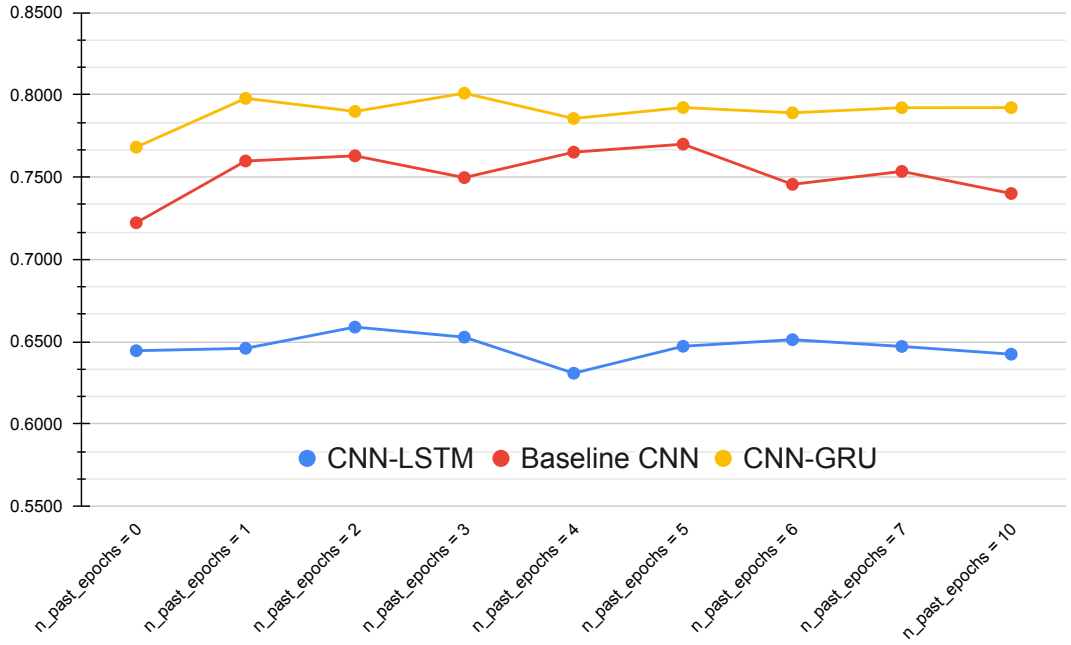


Figure 7.3: RQ 2 - TEST BA INTRA-DATASET RESULTS. *Balanced accuracy results across epochs. The graph shows the test BA as the parameter n_previous_epochs ranges from 0 to 10.*

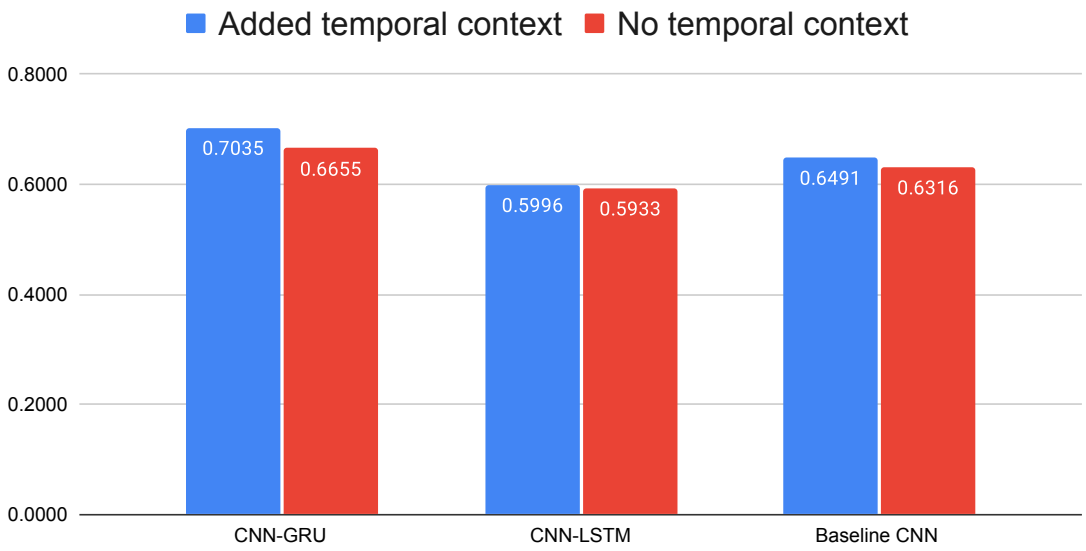


Figure 7.4: RQ 2 - TEMPORAL CONTEXT CROSS-DATASET RESULTS. *Comparison of temporal context training, obtained by averaging the BA score for parameter n_past_epochs ranging from 1 to 7, compared with baseline training (n_past_epochs = 0). We again make the comparison in a cross-dataset fashion. The graph shows that temporal context gives higher model performance over non-temporal context training for the CNN-GRU and baseline CNN architectures.*

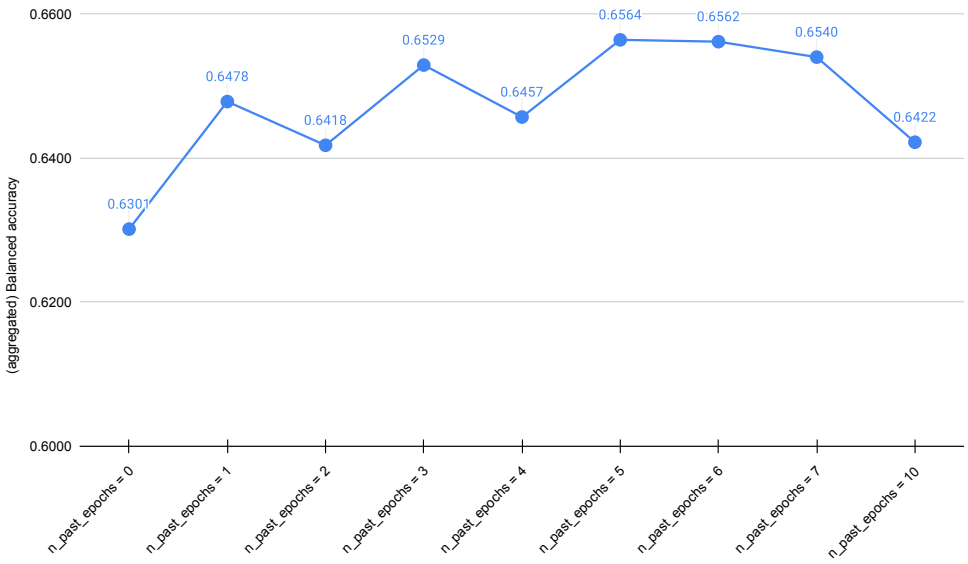


Figure 7.5: RQ 3 - AGGREGATED BALANCED ACCURACY RESULTS. *Cross-dataset BA aggregation across temporal context size (from 0 up to 10) for the BA scores obtained by all tested architectures. There is a positive correlation between the considered sequence length and the score of the models.*

7.3 RQ 3: The Longer Temporal Context, the Better (up to a threshold)

We test our model architectures for this experiment and concatenate previous sample instances to find the optimal sample length. As the graph in Figure 7.5 shows, there is a positive correlation between the considered sequence length and the BA score of the models. The aggregated cross-dataset BA peak is reached when the number of previous epochs concatenated is 5, for an aggregated BA score of 0.6564. After the peak, the BA score starts to decrease until we reach a lower score of 0.6422 when the number of concatenated samples is 10. There is a difference of 2.63 percentage points between the peak BA score and the score obtained when we do not concatenate any epoch to the data samples (n_past_epochs = 0) when we reach a BA score of 0.6301.

Another confirmation of such a result is visually represented using the confusion matrices in Figure 7.6, this time only analyzing the intra-dataset test score achieved by the CNN-GRU model, our best-performing architecture. Looking at the BA class difference for intra-dataset analysis on the MASS-SS3 subset, we see that as temporal context increases (parameter n_past_epochs going from 0 up to 10), the classification of stage N1 improves, going from a score of 0.57 when n_past_epochs = 1 up to 0.74 when n_past_epochs = 5.

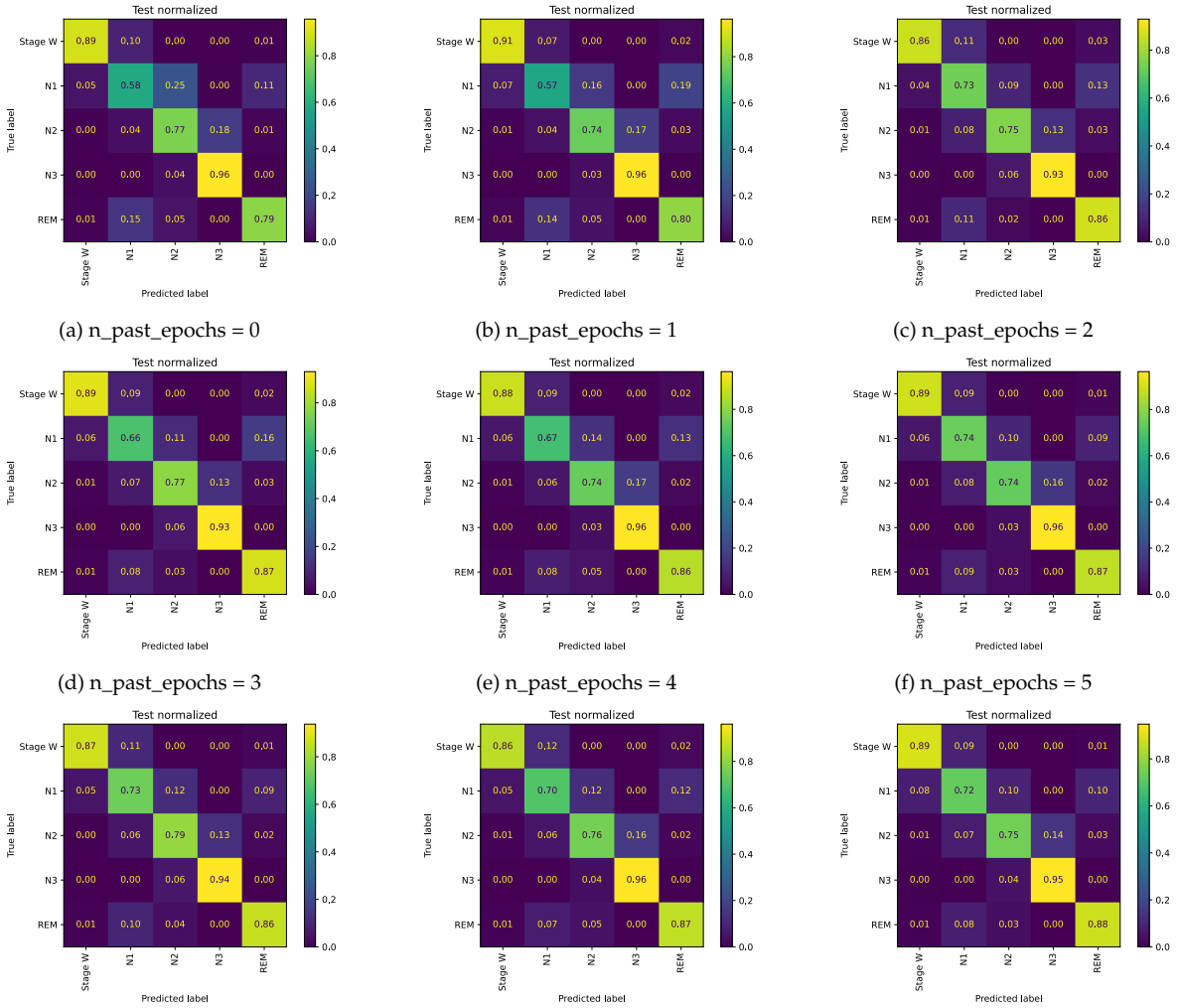
7.4 RQ 4: Weighted Cross-entropy Loss Works Better than Non-weighted Losses

On our best performing model, CNN-GRU, we have evaluated different choices of loss functions: The weighted CE loss, the focal loss, the MSE loss, and lastly, the KL loss. The results are visually shown in Figure 7.7.

The weighted CE loss performs better among the tested options, training the model with a BA value aggregated across datasets of 0.7220. The KL loss shows similar results, achieving a final aggregated value of 0.6972. Non-surprisingly, the MSE loss is the worst performer among the analyzed losses for our classification task, with a final aggregated value of 0.6651.

Figure 7.8 displays the results segmented into dataset protocols. One aspect that can be noted is the protocol-specific variance in each loss function setting and how it reflects on the model's BA. For the EDFxMASS-a protocol, the difference in BA between the top-performing loss function (Weighted CE) and the worst-performing loss (MSE loss) is almost 10%. For the EDFxMASS-d protocol, the losses do not seem to greatly affect the model's BA result.

To have a class-specific overview of the intra-dataset results, we analyze the confusion matrices obtained for the SC-EDF subset with higher BA results. We look at the BA results on the test set for n_past_epochs = 6 for each tested loss function. The confusion matrices are represented in Figure 7.9. In the confusion matrices, we see KL loss in Subfigure 7.9d is not able to classify the N1 class correctly, similar to the Focal loss and the MSE loss in Subfigure 7.9b and Subfigure 7.9c, respectively. Especially for the focal loss, the trained model cannot correctly classify stages N2 and N3. On the contrary, the CE loss achieves the highest test accuracy score for the N1 class.



(a) n_past_epochs = 0 (b) n_past_epochs = 1 (c) n_past_epochs = 2

(d) n_past_epochs = 3 (e) n_past_epochs = 4 (f) n_past_epochs = 5

(g) n_past_epochs = 6 (h) n_past_epochs = 7 (i) n_past_epochs = 10

Figure 7.6: RQ 3 - TEMPORAL CONTEXT - INTRA-DATASET ANALYSIS. *Intra-dataset analysis of experimental results on SS3 test subset (normalized) coming from the MASS dataset; As we increase the temporal context parameter on the best-performing model with GRU layer (parameter n_past_epochs going from 0 up to 10). The classification of class N1 improves as we increase the temporal context for training our model. The model misclassifies more often the N1 class as N2, and the N2 class as N3. The misclassification does not improve with added temporal context: This can be explained by the prevalence of N2 samples over N1. Overall, there is no class-specific difference when tuning temporal context in the MASS-SS3 dataset for the CNN-GRU model.*

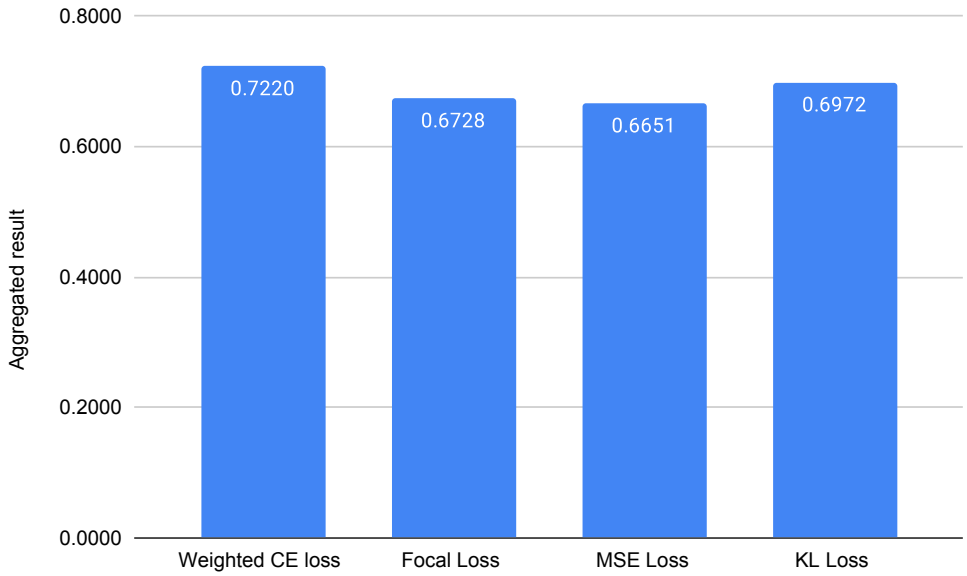


Figure 7.7: RQ 4 - Loss Functions Histogram Results. *Histogram representing the aggregated BA score for temporal context n_past_epochs = 6 and one bar for each tested loss function.*

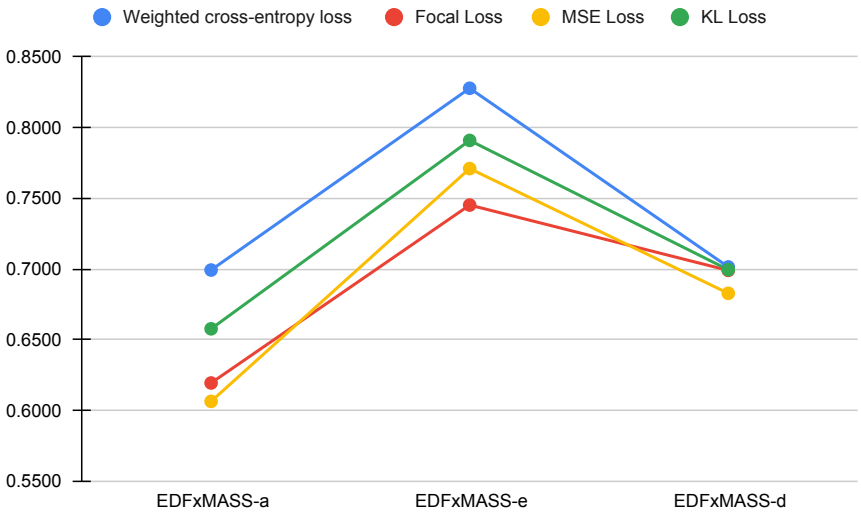
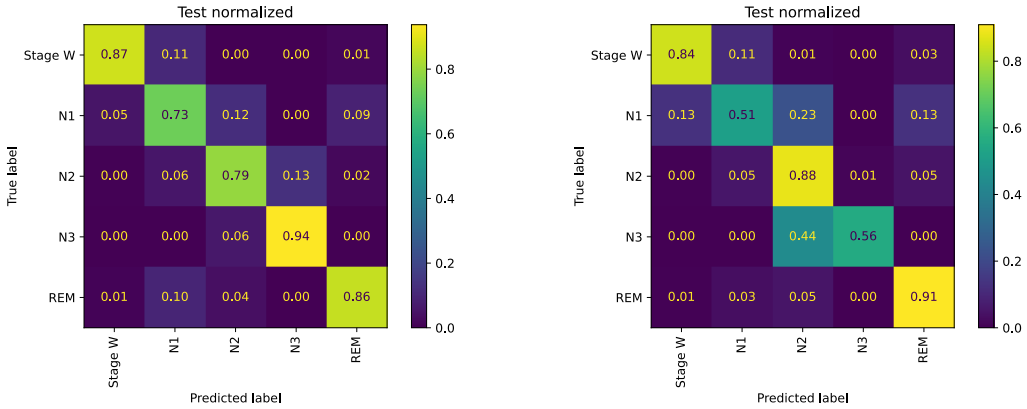
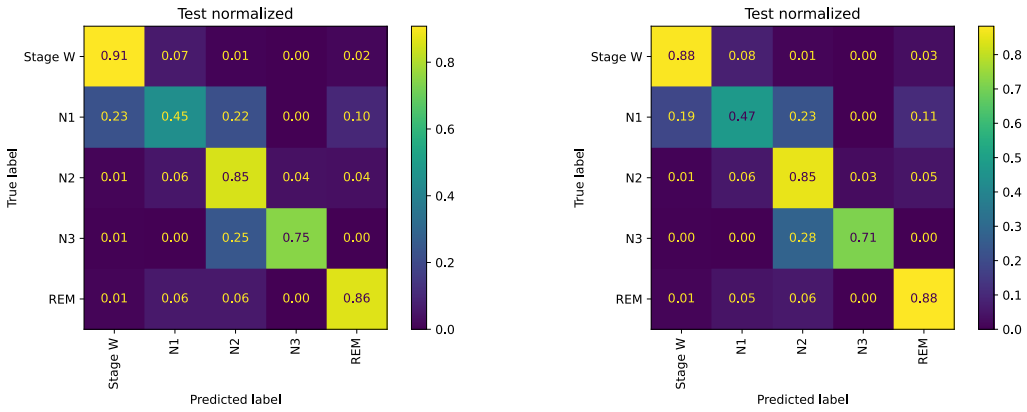


Figure 7.8: RQ 4 - Loss Functions Chart Results. *Balanced accuracy score for n_past_epochs = 6. For each tested loss function, we depict with a dot the BA performance for each dataset protocol (EDFxMASS-a, EDFxMASS-d, and EDFxMASS-e). The BA scores are generally higher for protocol EDFxMASS-e.*



(a) BA score while training with CE loss

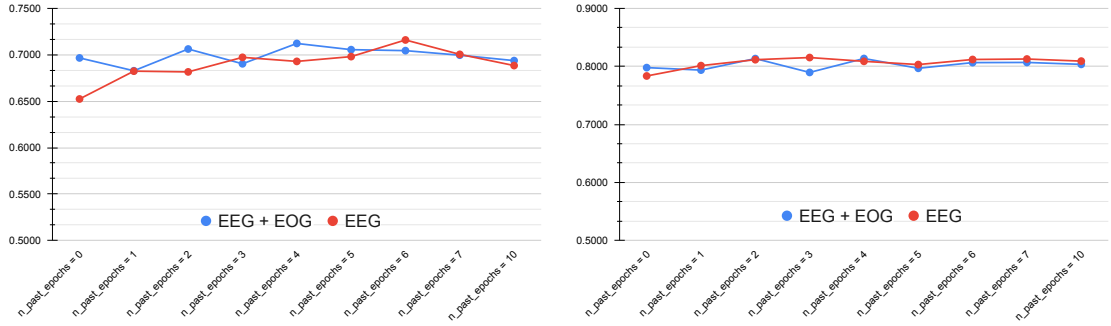
(b) BA score while training with Focal loss



(c) BA score while training with MSE loss

(d) BA score while training with KL loss

Figure 7.9: RQ 4 - TEST BA PER CLASS, VARYING LOSS. *For each tested loss function, we represent the confusion matrix showing the normalized test BA across classes, for n_past_epochs = 6 for each tested loss function.*



(a) EEG+EOG comparison with EEG (inter-dataset)

(b) EEG+EOG comparison with EEG (intra-dataset)

Figure 7.10: RQ 5 - MULTIVARIATE EEG AND EOG COMPARISON. *Subfigure (a) shows the intra-dataset results when adding an EOG channel in the data protocols (blue line) versus using the baseline 2-channeled EEG data. Subfigure (b) shows results for the inter-dataset analysis on the same multivariate channel comparison. We cannot see any improvement in BA for the databases we have used.*

7.5 RQ 5: Horizontal EOG Channel has No Tangible Effect on the Model Accuracy

For the final RQ, we want to test whether adding an EOG channel to our datasets improves the model classification accuracy. The cross-dataset and intra-dataset results on the CNN-GRU architecture can be visualized in Subfigures 7.10a and 7.10b, respectively. We cannot see any apparent improvement in BA when including a horizontal EOG channel in the databases we have used for intra- and inter-database analysis. There are slight variations in BA score as the number of concatenated epochs varies from 0 to 10.

Chapter 8

Discussion and Conclusion

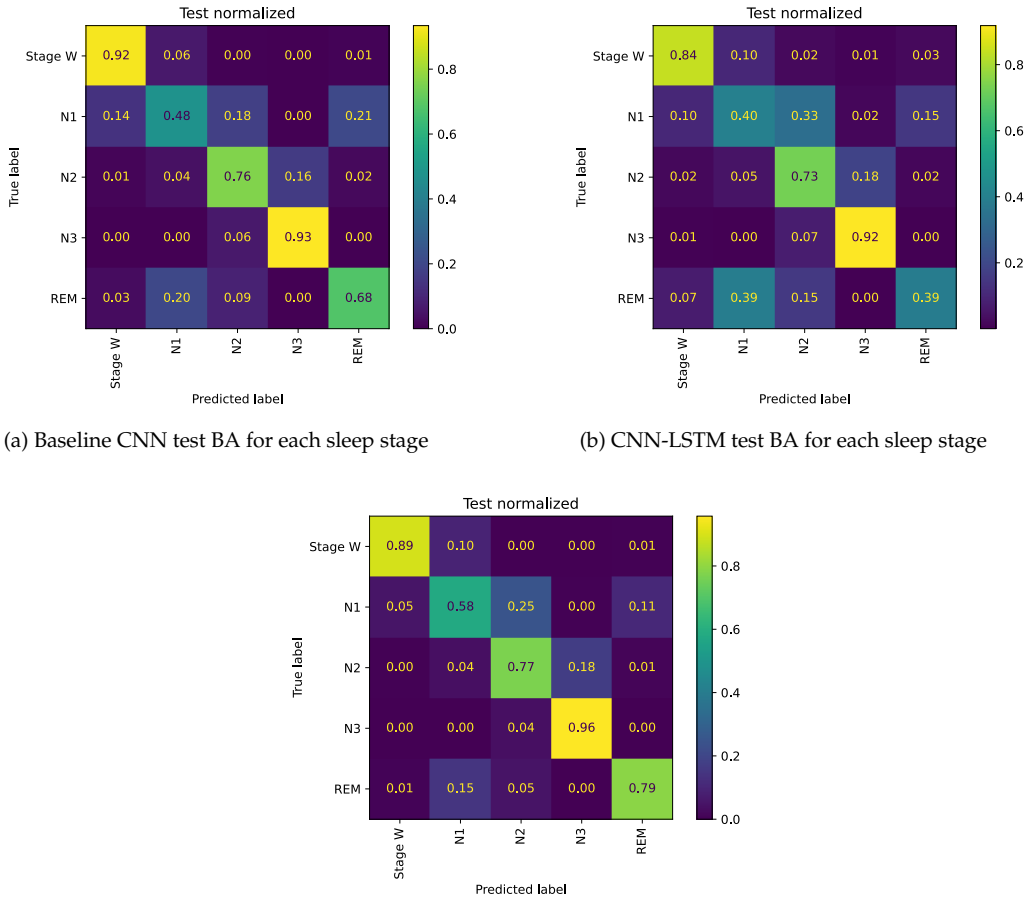
Addressing the limitations and challenges of automatic sleep stage classification systems requires ongoing research and development. Continued collaboration between sleep experts, engineers, and data scientists can help refine and improve automated systems to enhance their accuracy, reliability, and applicability in various clinical and research settings. In this finalizing chapter of our work, we want to interpret the results we got for our experiments by cross-analyzing them with both state-of-the-art deep learning models for sleep stage scoring and traditional manual scoring systems.

8.1 Experimental Findings

In this section, we want to give our interpretation based on the findings of the experiments.

Experiments on **RQ 1** showed the superior performance of the GRU layer compared with the LSTM layer for classification. This result can be attributed to numerous differences between the GRU and the LSTM layers, one among all: The LSTM layer is far more complex in terms of parameters than the GRU layer. Consequently, training an LSTM layer requires more attention to the hyperparameters and data required to achieve acceptable results and avoid overfitting. In the CNN-LSTM model experiment, overfitting seems to be one of the leading causes of such poor performance, as the graphs in Figure A.1 show: We represented the training and validation loss across epochs respectively for protocols EDFxMASS-a, EDFxMASS-d, and EDFxMASS-e for the CNN-LSTM model. The training and validation losses seem pretty high for all dataset protocols, getting as high as 0.8170 for the lowest validation loss achieved on cross-dataset EDFxMASS-d. To have an even broader overview of this phenomenon, we look at the performance difference across classes in the confusion matrix for each tested model on the MASS-SS3 test set. The results are displayed in Figure 8.1. The class scores show that the CNN model with the LSTM layer classifies the N3 stage almost as well as the other two architecture variants. However, looking at stages N1 and REM, we see a very stark difference between the LSTM and the GRU layers: The accuracy for LSTM drops drastically. The model cannot distinguish between class N1 and REM and often confounds.

When testing the models on temporal context for **RQ 2**, we again confirm the superiority of the CNN-GRU model configuration compared with the baseline CNN and the CNN-LSTM architectures. When comparing the GRU model with the baseline CNN, we get confirmation of the ability of recurrent units when handling temporal data, especially in the form of time series, compared with a simpler architecture developed by Chambon et al. (2018). The lower temporal context improvement for CNN-LSTM can be an additional flag that the model is overfitting: Adding more temporal context information is not helpful in this case. The baseline CNN model and the CNN model with the LSTM layer are less sensitive to training on longer temporal context



(a) Baseline CNN test BA for each sleep stage

(b) CNN-LSTM test BA for each sleep stage

(c) CNN-GRU test BA for each sleep stage

Figure 8.1: TEST BA PER CLASS - CONFUSION MATRICES. *For each model architecture, we represent the confusion matrix showing the normalized test BA for each sleep phase.*

sequences. Adding temporal context to training samples for the CNN with added GRU layer positively impacts the validation accuracy.

The intuition behind the observed differences in sensitivity to longer sequences and the impact of adding temporal context between the baseline CNN model and the model with added GRU layer can be explained by the nature of the architectures and their ability to capture temporal dependencies. The baseline CNN model implemented is primarily designed to operate on spatial features. It excels at extracting local patterns and spatial information from the input data. However, with explicit mechanisms to model temporal dependencies, the baseline CNN may be able to capture long-range and subtle temporal patterns in the data effectively. As a result, the model's performance degrades when trained on longer sequences due to the difficulty in maintaining temporal coherence. Adding a GRU layer to the baseline CNN model introduces an RNN component that handles sequential data.

Similarly to RQ 1 and RQ 2 experiments, we get a lower performance for the model with the LSTM layer classifier. Also, the model is not sensitive to temporal context, and accuracy does not vary when sample length changes. Following a similar approach, we want to analyze the same

phenomenon on an intra-dataset comparison. The results in Figure 7.3 tell a similar story to the cross-dataset results: For the CNN model with GRU unit and the baseline CNN model, there is some improvement in BA as the amount of temporal information in the dataset increases. The BA score flattens at some point or decreases for the baseline CNN model. The interpretation is analogous in this case.

The results in **RQ 3** show that more temporal context improves the final result, but up to a certain threshold: In this case, we argue that the relevance of previous sleep epochs decreases the further we concatenate back in time. Intuitively, temporal proximity is important, and we cannot feed infinitely long time series data to our model, which does not capture any additional information to predict the current sleep phase.

In the experiments for **RQ 4**, the weighted CE loss is the best performing: This finding suggests that the dataset used for training may be imbalanced, and the model could benefit from giving more weight to certain classes during the training process. By doing so, the model is more likely to focus on improving its performance on the underrepresented classes, leading to better overall accuracy. The MSE loss function is non-convex for multiclass classification. Thus, if a classification model is trained with the MSE loss function, it is not guaranteed to reach a minimum point: This can be one of the many reasons the MSE loss is not a good option for classification tasks, as the results have also demonstrated. The KL Loss performs similarly to the weighted CE loss function. This result makes sense as, under certain assumptions, the KL loss and the CE loss functions are asymptotically equal. While these loss functions provide competitive results, the weighted CE loss still outperforms it. Finally, we can justify the variability in the final accuracy score on the protocol-specific analysis (Figure 7.8) by the amount of training data in each protocol: EDFxMASS-a, having the smallest number of training epochs (only 27'314, compared with 122'278 for EDFxMASS-e and 36'339 for EDFxMASS-d) is the protocol on which our model has the highest variability in final BA score, and the lowest aggregated result overall. This indicates that loss functions are affected by the amount of training data fed to the network and that the MSE is the highest affected loss.

In the results of the experiments run for **RQ 5**, we find that the Horizontal EOG channel does not improve the model accuracy. We speculate that all information in the EOG data is redundant with what the model has already learned from the EEG channels data. We could also attribute the result to underfitting: By adding an additional channel with more temporal information, the model would not fit properly, and thus no improvement in balanced accuracy can be seen. The slight variability we noticed across temporal context length could be due to chance. Being the variability in the BA score of small entity, we consider it as random and impute it to our databases' relatively small size.

8.2 State-of-the-art Comparison

We want to give an overview of our model architectures in the general task framework. To compare our models with the state-of-the-art, we use the simple accuracy score as being one of the most widely used metrics in general and being easily comparable and generally correlated with the other metrics we have used. Specifically, we compare our best result for each tested model architecture with the ones developed by Tsinalis et al. (2016), Supratak et al. (2017), Phan et al. (2021) and Chambon et al. (2018). The model by Tsinalis et al. (2016) comprises Stacked Sparse Autoencoders. The primary distinction between stacked autoencoders and conventional neural networks is using unlabelled data (data without class labels) for layer-wise pre-training before fine-tuning the network. Iterative optimization and the backpropagation method are used to train the autoencoders. The DeepSleepNet model created by Supratak et al. (2017) uses bidirectional-LSTMs and two CNNs with various filter sizes at the first layers. The bidirectional-LSTMs are

trained to encode temporal information, such as sleep stage transition rules, into the model. At the same time, the CNNs can be taught to develop filters to extract time-invariant features from raw single-channel EEG. They use a two-step training strategy that efficiently trains the model from beginning to end using backpropagation while guarding against class imbalance issues. The sequence-to-sequence network architecture for autonomous sleep staging created by Phan et al. (2021) and defined XSleepNet can simultaneously learn from raw signal and time-frequency input. The network design supports two network streams, one for each input view. The XSleepNet model has been built to be robust to training data size, complimentary between the individual network streams, and aware of its network streams' generalization and overfitting behavior. The network can be trained to encourage learning on the generalizing network stream while discouraging learning on the overfitting one. XSleepNet1 and XSleepNet2 are two models created to approximate generalization and overfitting measures on classification branches of the network and produce the appropriate weights for gradient blending.

We want to highlight the different settings we start from: Our models have been trained using parameters that differ from that of the analyzed state-of-the-art models. We also used a multivariate approach by including Fpz-Cz and Pz-Oz channels in our setting. At the same time, most of the other studies we analyzed consider the EEG bipolar channel Fpz-Cz only. The temporal context concatenation we defined in Section 5.2 is also a preprocessing technique that distinguishes our setting from comparable sleep stage classification studies.

The results are visually represented in Figure 8.2. In Table 8.1, we compare the results of our model architectures for two subsets we have used: Night SS3 recorded in the MASS dataset and subset SC-EDF in Sleep-EDF dataset. Analogously to the results observed in the experiments, our best-performing model is the CNN-GRU model. When considering the results we obtained on dataset SC-EDF using the CNN-GRU model, the accuracy is comparable to the state-of-the-art results obtained using the autoencoder, sequence-to-sequence, and CNN models. We get an accuracy result of 0.8268, which is slightly higher than the result obtained by the Stacked Sparse Autoencoders model (Tsinalis et al., 2016), DeepSleepNet (Supratak et al., 2017) and on XSleepNet1 (Phan et al., 2021). Considering the BA results, the baseline CNN model developed by Chambon et al. (2018) and the CNN model with added LSTM recurrent layer cannot achieve comparable performance to the rest of the results. These results point in the same direction as the findings of the previous experiments. The results obtained on the MASS-SS3 subset indicate a different scenario: On this dataset, we see a relatively higher performance for the DeepSleepNet model developed by Supratak et al. (2017). Our best model, CNN-GRU, achieves 0.8108 as an accuracy value, still relatively higher than the model developed by Chambon et al. (2018).

8.3 Cohen's κ Comparison Across Protocols

Cohen's κ provides a straightforward interpretation by indicating how much our classifier outperforms a random classifier based on class frequencies. By calculating the inter-rater reliability metric for a specific rating protocol (e.g., AASM rating) and the Linear κ from our model, we can then compare these metrics to determine whether implementing our model in a real-world setting brings benefits to the final annotation outcome. It is essential to note the following remarks: If the inter-rater reliability metric is low, the human raters have significant disagreements among themselves. In this case, if the model achieves a higher Linear κ than the inter-rater reliability, it performs better than the average agreement between human raters, bringing real-world advantage to the task. On the opposite side, if the inter-rater reliability metric is high, indicating a high level of agreement among human raters, and the model's Linear Weighted κ is lower than the inter-rater reliability, it suggests that the model may not perform as well as the consensus among human raters, therefore requiring additional work and analysis.

Table 8.1: STATE-OF-THE-ART ACCURACY RESULTS. *Comparison between state-of-the-art deep learning models in sleep phase classification and the model architectures we implemented for both SC-EDF subset (6 model comparisons) and MASS-SS3 subset (3 model comparisons). The column "# Channels used" indicates which channels have been included in the subset for training and testing. The column "# Past Epochs" indicates how many previous epochs have been concatenated to the current epoch in the temporal context setting defined in Section 5.2 of the Approach chapter.*

Model (study) name	# Channels used	# Past epochs	Accuracy
	SC-EDF		
CNN-GRU (our model)	**(Fpz-Cz, Pz-Oz)**	**7**	**0.8268**
Stacked Sparse Autoencoders (Tsinalis et al., 2016)	Fpz-Cz	0	0.8200 (mean)
Baseline CNN (Chambon et al., 2018)	(Fpz-Cz, Pz-Oz)	3	0.7653
DeepSleepNet (Supratak et al., 2017)	Fpz-Cz	0	0.8200
XSleepNet2 (Phan et al., 2021)	Fpz-Cz	0	**0.8390**
XSleepNet1 (Phan et al., 2021)	Fpz-Cz	0	0.8220
	MASS-SS3		
CNN-GRU (our model)	**(Fpz-Cz, Pz-Oz)**	**3**	**0.8108**
DeepSleepNet (Supratak et al., 2017)	F4-EOG (Left)	0	**0.8620**
Baseline CNN (Chambon et al., 2018)	(Fpz-Cz, Pz-Oz)	1	0.7796

Table 8.2: LINEAR WEIGHTED κ - INTER-DATASET RESULTS. *Linear weighted κ measure of model agreement averaged across protocols and respective judgment. This analysis was conducted on a cross-database setting (EDFxMASS).*

Protocol	Linear Weighted Kappa	Judgment
EDFxMASS-a	0.5240	Moderate agreement
EDFxMASS-d	0.6234	Substantial agreement
EDFxMASS-e	**0.7681**	**Substantial agreement**

To evaluate our model results on intra- and cross-dataset settings, we select the best model architecture, CNN-GRU, using the best configuration across protocols (Weighted CE loss function and n_past_epochs = 6). The results of the experimental trials on the cross-dataset setting are shown in Table 8.2. The results of our trials on the intra-dataset analysis are reported in Table 8.3. In the cross-dataset results, we notice the difference in κ score across protocols: When the model is trained on the SC-EDF subset and tested on ST-EDF and MASS-SS3 subsets (EDFxMASS-e case), the κ score gives a substantial agreement judgment of 0.7681. This result can be justified by the fact that the SC-EDF subset, of size 122'278 epochs, on which the model is trained, is significantly larger than training subsets ST-EDF and SS3, which respectively have 27'314 and 36'339 training epochs.

The results on intra-dataset results are coherent with the inter-dataset analysis: The degree of agreement between our CNN-GRU architecture and the test set is highest for the SC-EDF subset (the training set used in protocol EDFxMASS-e) and lowest for ST-EDF subset (the training set used in protocol EDFxMASS-a). As expected, the absolute values are generally higher across all subsets in the intra-dataset case. In this case, the test set comes from the same subset used for training the model. Therefore, we expect the general model performance to be higher. In all datasets, we reach a substantial agreement judgment. Overall, the results we have found here are coherent with the balanced accuracy scores obtained in the previous experiments. EDFxMASS-a is the protocol where our model has the most significant difficulties when classifying sleep stages. It is essential to note that the different subsets used to create the cross-dataset training and testing splits are statistically different; This can impact the final model classification performance. The data size is a factor to consider when training neural network models. In Table A.5, we present the statistics on each dataset size for the cross-subset protocols (number of epochs and samples in

Table 8.3: LINEAR WEIGHTED κ - INTRA-DATASET RESULTS. *Linear weighted κ measure of model agreement on the test sets of our subsets and respective judgment. This analysis was conducted in an intra-database setting, training, and testing on the same subset.*

Subset	Linear Weighted Kappa	Judgment
ST-EDF	0.6671	Substantial agreement
MASS-SS3	0.7899	Substantial agreement
SC-EDF	**0.7818**	**Substantial agreement**

each subset).

8.4 Inter-rater Agreement Comparison

When analyzing such experimental results, we need a real-world comparison to understand how automated systems powered by neural network engines work against manual annotation of sleep phases. Automatizing the annotation procedure brings advantages in terms of time and human effort, reducing both to a minimum. We want to see the same improvement also in terms of performance in the final results of our work. To evaluate the agreement between different human raters and compare it to the performance of the best-performing algorithm for sleep classification, we need a comparable measure of the algorithm's performance. Therefore, we will utilize Cohen's κ measure for our model.

We want to compare our results with the inter-rater reliability in sleep stage scoring using the AASM and R&K manuals. For each manual scoring study we take into consideration and compare to our CNN-GRU model, we highlight the differences in terms of sleep scoring rules, the number of EEGs used to agree, the scorer's characteristics, the number of epochs and the overall Cohen's κ.

The results can be visualized in Table 8.4. The Cohen's κ results differ widely across subsets, from 0.5079 up to 0.7754 in the inter-dataset results and 0.6671 up to 0.7818 in the intra-dataset results. For intra-dataset protocol EDFxMASS-e and subsets SC-EDF and MASS-SS3, the accuracy results are higher than the inter-rater agreement achieved by studies of Deng et al. (2020), Shambroom et al. (2012) and Norman et al. (2000). On the contrary, manual scoring agreement results achieved by Kubicki et al. (1989) and Schaltenbrand et al. (1996) far surpass the Cohen's κ result we have achieved across all protocols and subsets, both in the inter- and intra-dataset analysis. The result can be due to the used scoring rule, which for both studies is 7-class and 6-class R&K, while for our study, we have based our analysis on the 5-class AASM scoring rules. Also, the scorer's attributes might have played a role in the recorded Cohen's κ result: For the inter-rater agreement study by Kubicki et al. (1989), there are only two very experienced coders, while for the study by Schaltenbrand et al. (1996), they have employed two coders with non-reported experience.

8.5 Conclusion and Future Work

We develop an experimental architecture protocol and evaluation procedure to tackle the automatic sleep phase classification task and compare its reliability to manual sleep scoring results. First, we test whether adding an RNN layer benefits the model's prediction ability. Then, we include additional context by concatenating training samples to prove whether our models can learn from them. We run tests with different loss functions to compare weighted loss functions

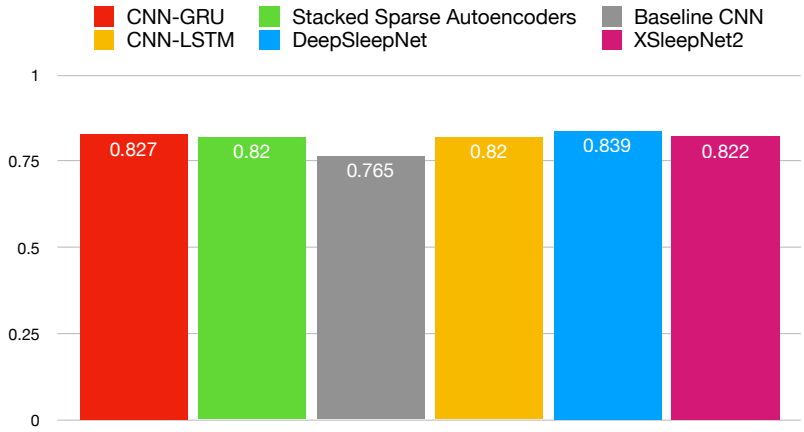
with more straightforward solutions and to test their effect on the final balanced accuracy of the best-performing model. Lastly, we test the effect of adding an additional EOG channel in the sample epochs on the final accuracy of our best architecture.

We thoroughly analyzed our results to draw meaningful conclusions from the previous sections. Firstly, it is essential to reiterate the significance of PSG exams as a primary examination for sleep analysis. Traditionally, data obtained from PSG analysis is classified manually by expert technicians. Our investigation focused on automatic sleep stage classification and the impact of including the temporal context in the samples. We found that the CNN model with added GRU classifier can effectively extract contextual information from training samples, improving generalizability. We also found that increasing the temporal context by providing an extended sequence length benefits the final model's ability to correctly predict the sleep classes up to a certain number of concatenated previous epochs. The evidence on the intra-dataset analysis demonstrates that our model performs equally to some of the analyzed manual sleep stage scoring conducted by expert technicians in terms of Cohen's κ metric. Empirical results consistently support our model's ability to accurately identify and classify sleep stages, just as a human expert would be able to do. These results serve as proof of concept for the effectiveness of our neural network-based approach, which does not require any supervision or hand-crafted feature to train. We showed that a CNN with an added GRU layer could achieve higher or comparable performance in automatic sleep stage scoring compared to alternative state-of-the-art approaches. Our work shows that end-to-end training in CNNs is effective in terms of sleep stage scoring performance and can potentially have applications in other biosignal-based classification problems (for example, by using EMG and ECG channels).

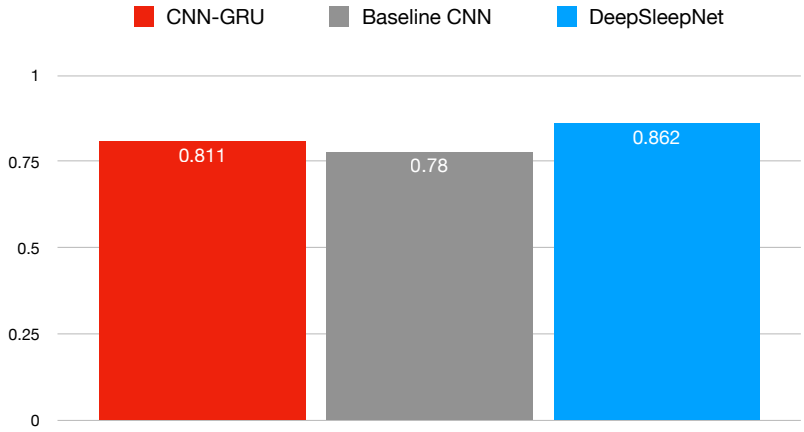
Future Work

Large sleep staging datasets still need to be collected and made public. The investigated studies and the work we have developed utilized custom dataset protocols: We used a combination of Sleep-EDF and MASS subsets to achieve model generalizability. However, the available data are still limited. Including semantic data annotation, data sharing, and online applications would promote a better understanding of sleep physiology by making readily accessible large sleep datasets. This could facilitate international and inter-disciplinary collaboration in sleep medicine, and the final goal of sleep disorders detection would be better achievable.

A final observation is that there needs to be more consensus on the amount and frequency of filters employed for preprocessing the raw PSG signal. More research on the area would improve the quality of the data fed into the models and improve the final classification result.



(a) state-of-the-art comparison on SC-EDF subset



(b) state-of-the-art comparison on MASS-SS3 subset

Figure 8.2: STATE-OF-THE-ART INTRA-DATASET ACCURACY RESULTS. *Barplot comparing the model architectures result in terms of intra-dataset accuracy. Subfigure (a) shows results on the SC-EDF subset, and Subfigure (b) shows the accuracy results on the MASS-SS3 subset. Our CNN-GRU architecture achieves comparable performance on the SC-EDF test set, while on the MASS-SS3 subset, it slightly underperforms compared with the DeepSleepNet model.*

Table 8.4: MANUAL SCORING STUDIES - κ COMPARISON. *Linear weighted κ measure of model agreement averaged across protocols and respective judgment. This analysis was conducted in a cross-database setting (EDFxMASS). The first section of the table contains the results of the CNN-GRU model on subsets SC-EDF (a), MASS-SS3 (b), and ST-EDF (c). The second section of the table contains results for cross-protocols EDFxMASS-a (d), EDFxMASS-d (e), and EDFxMASS-e (f). The last section of the table contains the results of the manual scoring studies.*

	Rule/Number of Stages	**EEGs**	**Scorers/Scorers' Characteristics**	**Test Epochs**	**Patient Characteristics**	**Cohen's κ**
CNN-GRU model						
(a) SC-EDF	AASM/5	2	-	34'559	Healthy	0.7818
(b) MASS-SS3	AASM/5	2	-	11'377	Healthy	0.7899
(c) ST-EDF	AASM/5	2	-	7973	Healthy	0.6671
(d) EDFxMASS-a	AASM/5	2	-	388'430	Healthy	0.5079
(e) EDFxMASS-d	AASM/5	2	-	375'291	Healthy	0.6133
(f) EDFxMASS-e	AASM/5	2	-	82'741	Healthy	0.7754
(Deng et al., 2020) 2020/China	AASM/5	6	7/at least 2y of AASM scoring	37'642	Healthy, Obstructive Sleep Apnea (OSA)	0.7537
(Shambroom et al., 2012) 2012, U.S.A.	R&K/4	6	2/trained in a research setting	19'556	Healthy	0.7370
(Kubicki et al., 1989) 1989/U.S.A	R&K/7	1	2/very experienced	13'850	Healthy	0.8584
(Schaltenbrand et al., 1996) 1996/France	R&K/6	2	2/Not reported	21'138	Healthy	0.8364
(Norman et al., 2000) 2000/U.S.A.	R&K/5	2	5/average experience 13.4y, range 7–24 y	53'735	Healthy, OSA	0.6263

Appendix A

Attachments

Listed below are the tables of experiments run to get the final results.

A.1 Cross-dataset results

Table A.1: CNN-GRU - CROSS-DATASET RESULTS. *Experimental results obtained for CNN-GRU model, varying the number of previous epochs parameter n_past_epochs. The results displayed are inter-datasets, that is, for each protocol (EDFxMASS-a, EDFxMASS-d, EDFxMASS-e). The training and testing datasets are different from each other.*

CNN-GRU	**EDFxMASS-a**	**EDFxMASS-d**	**EDFxMASS-e**	**BA**
epochs	209141	196002	82741	
n_past_epochs = 0	0.6167	0.6828	0.7477	0.6655
n_past_epochs = 1	0.6561	0.6823	0.8057	0.6920
n_past_epochs = 2	0.6663	0.6579	0.8090	0.6871
n_past_epochs = 3	0.6849	0.6697	0.8202	0.7017
n_past_epochs = 4	0.6676	0.6982	0.7989	0.7022
n_past_epochs = 5	0.6657	0.7160	0.8070	0.7099
n_past_epochs = 6	0.6994	0.7015	0.8279	0.7220
n_past_epochs = 7	0.6763	0.6925	0.8324	0.7093
n_past_epochs = 10	0.6662	0.6834	0.8041	0.6965

Table A.2: CNN-LSTM - CROSS-DATASET RESULTS. *Experimental results obtained for CNN-LSTM model, varying the number of previous epochs parameter n_past_epochs. The results displayed are inter-datasets, that is, for each protocol (EDFxMASS-a, EDFxMASS-d, EDFxMASS-e). The training and testing datasets are different from each other.*

CNN-LSTM	EDFxMASS-a	EDFxMASS-d	EDFxMASS-e	BA
epochs	209141	196002	82741	
n_past_epochs = 0	0.5814	0.5815	0.6515	0.5933
n_past_epochs = 1	0.5943	0.5922	0.6530	0.6034
n_past_epochs = 2	0.5930	0.5875	0.6645	0.6029
n_past_epochs = 3	0.5837	0.5986	0.6586	0.6024
n_past_epochs = 4	0.5850	0.5924	0.6231	0.5944
n_past_epochs = 5	0.5846	0.5919	0.6493	0.5985
n_past_epochs = 6	0.5823	0.5849	0.6566	0.5959
n_past_epochs = 7	0.5846	0.5886	0.6624	0.5994
n_past_epochs = 10	0.5724	0.5850	0.6504	0.5907

Table A.3: BASELINE CNN - CROSS-DATASET RESULTS. *Experimental results obtained for the baseline CNN model, varying the number of previous epochs parameter n_past_epochs. The results displayed are inter-datasets, that is, for each protocol (EDFxMASS-a, EDFxMASS-d, EDFxMASS-e), The training and testing datasets are different from each other.*

(Chambon et al., 2018)	EDFxMASS-a	EDFxMASS-d	EDFxMASS-e	BA
epochs	209141	196002	82741	
n_past_epochs = 0	0.6320	0.5935	0.7207	0.6316
n_past_epochs = 1	0.6195	0.6282	0.7676	0.6481
n_past_epochs = 2	0.6030	0.6098	0.7771	0.6353
n_past_epochs = 3	0.6304	0.6278	0.7795	0.6546
n_past_epochs = 4	0.6351	0.5884	0.7774	0.6405
n_past_epochs = 5	0.6349	0.6385	0.7796	0.6609
n_past_epochs = 6	0.6545	0.6125	0.7307	0.6505
n_past_epochs = 7	0.6524	0.6118	0.7543	0.6534
n_past_epochs = 10	0.6492	0.5905	0.7306	0.6394

Table A.4: LOSS FUNCTION - CROSS-DATASET ANALYSIS. *Experimental results obtained using the CNN-GRU model and changing the loss function parameter, namely, using the Weighted cross-entropy loss, Focal Loss, MSE Loss, and KL Loss alternatively. The results are aggregated across data protocols (EDFxMASS-a, EDFxMASS-d, and EDFxMASS-e).*

	EDFxMASS-a	EDFxMASS-d	EDFxMASS-e	Aggregated BA
epochs	209141	196002	82741	
Weighted CE loss	0.6994	0.7015	0.8279	0.7220
Focal Loss	0.6195	0.6991	0.7453	0.6728
MSE Loss	0.6065	0.6829	0.7711	0.6651
KL Loss	0.6578	0.6996	0.7910	0.6972

Table A.5: NUMBER OF 30S EPOCHS PER TEST PROTOCOL SUBSET - CROSS-DATASET. *Table containing the testing subsets of each cross-dataset protocol (EDFxMASS-a, EDFxMASS-d and EDFxMASS-e). The last two columns indicate the sum between the training and validation epochs (Train + valid) and the number of test epochs in the subsets, respectively. It is important to note how the numbers are quite similar for protocols EDFxMASS-a and EDFxMASS-d but widely different to protocol EDFxMASS-e, which only has 82'741 training+validation epochs and 19'350 test epochs.*

EDFxMASS-a

	edf_sc_train	edf_sc_validation	edf_sc_test	mass_ss3_train	mass_ss3_validation	mass_ss3_test	Train + valid	Test
sample number	28	8	8	38	12	12	86	20
sleep epochs	27314	7487	7973	36339	11601	11377	82741	19350

EDFxMASS-d

	edf_sc_train	edf_sc_validation	edf_sc_test	edf_st_train	edf_st_validation	edf_st_test	Train + valid	Test
sample number	97	28	28	28	8	8	161	36
sleep epochs	122278	38923	34559	27314	7487	7973	196002	42532

EDFxMASS-e

	edf_st_train	edf_st_validation	edf_st_test	mass_ss3_train	mass_ss3_validation	mass_ss3_test	Train + valid	Test
sample number	97	28	28	38	12	12	175	40
sleep epochs	122278	38923	34559	36339	11601	11377	209141	45936

A.2 Intra-dataset results

Table A.6: AGGREGATED BA FOR CNN-GRU MODEL - INTRA-DATASET RESULTS. *Experimental results obtained for baseline CNN-GRU model, varying the number of previous epochs parameter n_past_epochs.*

CNN-GRU	ST-EDF	SC-EDF	SS3	Test BA
test samples	8	28	12	
test epochs	7973	34559	11377	
n_past_epochs = 0	0.6800	0.7650	0.8400	0.7683
n_past_epochs = 1	0.7400	0.7911	0.8594	0.7980
n_past_epochs = 2	0.7200	0.7850	0.8543	0.7900
n_past_epochs = 3	0.7500	0.7968	0.8500	0.8011
n_past_epochs = 4	0.7400	0.7723	0.8584	0.7857
n_past_epochs = 5	0.7300	0.7874	0.8512	0.7924
n_past_epochs = 6	0.7500	0.7782	0.8498	0.7892
n_past_epochs = 7	0.7600	0.7792	0.8546	0.7923
n_past_epochs = 10	0.7439	0.7837	0.8523	0.7923

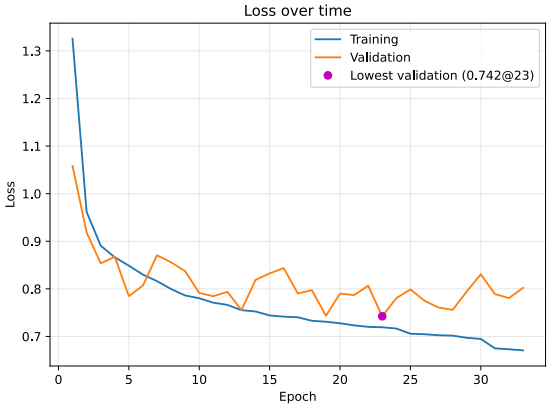
Table A.7: AGGREGATED BA FOR CNN-LSTM MODEL - INTRA-DATASET RESULTS. *Experimental results obtained for baseline CNN-LSTM model, varying the number of previous epochs parameter n_past_epochs.*

CNN-LSTM	ST-EDF	SC-EDF	SS3	Test BA
test samples	8	28	12	
test epochs	7973	34559	11377	
n_past_epochs = 0	0.6400	0.6437	0.6500	0.6445
n_past_epochs = 1	0.6500	0.6404	0.6600	0.6460
n_past_epochs = 2	0.6500	0.6573	0.6699	0.6589
n_past_epochs = 3	0.6400	0.6529	0.6612	0.6528
n_past_epochs = 4	0.6200	0.6204	0.6700	0.6308
n_past_epochs = 5	0.6400	0.6447	0.6600	0.6472
n_past_epochs = 6	0.6500	0.6446	0.6721	0.6512
n_past_epochs = 7	0.6500	0.6440	0.6544	0.6471
n_past_epochs = 10	0.6300	0.6461	0.6398	0.6424

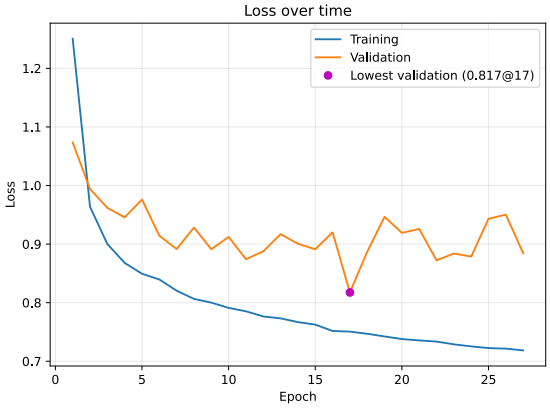
Table A.8: AGGREGATED BA FOR BASELINE CNN MODEL - INTRA-DATASET RESULTS. *Experimental results obtained for baseline CNN model, varying the number of previous epochs parameter n_past_epochs.*

Baseline CNN	**ST-EDF**	**SC-EDF**	**SS3**	**Test BA**
test samples	8	28	12	
test epochs	7973	34559	11377	
n_past_epochs = 0	0.7000	0.7228	0.7365	0.7223
n_past_epochs = 1	0.7200	0.7615	0.7829	0.7598
n_past_epochs = 2	0.7100	0.7730	0.7697	0.7630
n_past_epochs = 3	0.7200	0.7653	0.7233	0.7498
n_past_epochs = 4	0.7200	0.7700	0.7826	0.7652
n_past_epochs = 5	0.7300	0.7717	0.7932	0.7701
n_past_epochs = 6	0.6900	0.7457	0.7846	0.7457
n_past_epochs = 7	0.7140	0.7499	0.7924	0.7535
n_past_epochs = 10	0.6764	0.7391	0.7877	0.7401

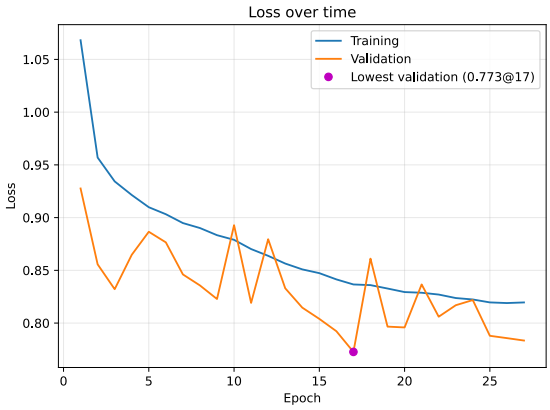
A.3 CNN-LSTM Model Loss Diagnostics



(a) CNN-LSTM loss on EDFxMASS-a protocol



(b) CNN-LSTM loss on EDFxMASS-d protocol



(c) CNN-LSTM loss on EDFxMASS-e protocol

Figure A.1: RQ 1 - Loss per Epoch Across Protocols. *The figure shows the training and validation losses across epochs for protocol EDFxMASS-a (A.1a), EDFxMASS-d (A.1b) and EDFxMASS-e (A.1c). The training loss shows how well the model fits training data, while the validation loss shows how well it matches new data.*

List of Figures

List of Tables

Bibliography

Belkhiria, C., Boudir, A., Hurter, C., and Peysakhovich, V. (2022). EOG-based human-computer interface: 2000–2020 review. *Sensors*, 22(13):4914.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478. Springer.

Berry, R. B., Brooks, R., Gamaldo, C., Harding, S. M., Lloyd, R. M., Quan, S. F., Troester, M. T., and Vaughn, B. V. (2017). Aasm scoring manual updates for 2017 (version 2.4).

Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE.

Bulling, A., Ward, J. A., Gellersen, H., and Tröster, G. (2010). Eye movement analysis for activity recognition using electrooculography. *IEEE transactions on pattern analysis and machine intelligence*, 33(4):741–753.

Carley, D. W. and Farabi, S. S. (2016). Physiology of sleep. *Diabetes spectrum: a publication of the American Diabetes Association*, 29(1):5.

Chambon, S., Galtier, M. N., Arnal, P. J., Wainrib, G., and Gramfort, A. (2018). A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(4):758–769.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Colarusso, P., Kidder, L. H., Levin, I. W., and Neil Lewis, E. (1999). Raman and infrared microspectroscopy. In Lindon, J. C., editor, *Encyclopedia of Spectroscopy and Spectrometry*, pages 1945–1954. Elsevier, Oxford.

Deng, Y., Liu, W., Liu, K., Fang, Y.-Y., Shang, J., Zhou, L., Wang, K., Leng, F., Wei, S., Chen, L., et al. (2020). Clinical characteristics of fatal and recovered cases of coronavirus disease 2019 in wuhan, china: a retrospective study. *Chinese medical journal*, 133(11):1261–1267.

Dong, H., Supratak, A., Pan, W., Wu, C., Matthews, P. M., and Guo, Y. (2017). Mixed neural network approach for temporal sleep stage classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(2):324–333.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

Gheisari, S., Shariflou, S., Phu, J., Kennedy, P. J., Agar, A., Kalloniatis, M., and Golzan, S. M. (2021). A combined convolutional and recurrent neural network for enhanced glaucoma detection. *Scientific reports*, 11(1):1945.

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):e215–e220.

Gratton, G. (1998). Dealing with artifacts: The EOG contamination of the event-related brain potential. *Behavior Research Methods, Instruments, & Computers*, 30(1):44–53.

Hafner, M., Romanelli, R. J., Yerushalmi, E., and Troxel, W. M. (2023). *The societal and economic burden of insomnia in adults: An international study*. RAND Corporation, Santa Monica, CA.

Hasan, M. J., Shon, D., Im, K., Choi, H.-K., Yoo, D.-S., and Kim, J.-M. (2020). Sleep state classification using power spectral density and residual neural network with multichannel EEG signals. *Applied Sciences*, 10(21):7639.

Herwig, U., Satrapi, P., and Schönfeldt-Lecuona, C. (2003). Using the international 10-20 EEG system for positioning of transcranial magnetic stimulation. *Brain topography*, 16:95–99.

Iber, C. (2007). The AASM manual for the scoring of sleep and associated events: Rules. *Terminology and Technical Specification*.

Imtiaz, S. A. and Rodriguez-Villegas, E. (2014). A low computational cost algorithm for REM sleep detection using single channel EEG. *Annals of biomedical engineering*, 42:2344–2359.

Jiang, X., Bian, G.-B., and Tian, Z. (2019). Removal of artifacts from EEG signals: a review. *Sensors*, 19(5):987.

Karpiel, I., Kurasz, Z., Kurasz, R., and Duch, K. (2021). The influence of filters on EEG-ERP testing: Analysis of motor cortex in healthy subjects. *Sensors*, 21(22):7711.

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

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *Cornell University*.

Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Krakovská, A. and Mezeiová, K. (2011). Automatic sleep scoring: A search for an optimal combination of measures. *Artificial intelligence in medicine*, 53(1):25–33.

Krieger, A. C. and Lee-Chiong, T. (2017). *Social and economic dimensions of sleep disorders*, volume 12. Elsevier.

Kubicki, S., Höller, L., Berg, I., Pastelak-Price, C., and Dorow, R. (1989). Sleep EEG evaluation: a comparison of results obtained by visual scoring and automatic analysis with the oxford sleep stager. *Sleep*, 12(2):140–149.

Lan, K.-C., Chang, D.-W., Kuo, C.-E., Wei, M.-Z., Li, Y.-H., Shaw, F.-Z., and Liang, S.-F. (2015). Using off-the-shelf lossy compression for wireless home sleep staging. *Journal of neuroscience methods*, 246:142–152.

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

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

Lee, Y. J., Lee, J. Y., Cho, J. H., and Choi, J. H. (2022). Interrater reliability of sleep stage scoring: a meta-analysis. *Journal of Clinical Sleep Medicine*, 18(1):193–202.

Li, H., Peng, C., and Ye, D. (2015). A study of sleep staging based on a sample entropy analysis of electroencephalogram. *Bio-medical materials and engineering*, 26(s1):S1149–S1156.

Liang, S.-F., Kuo, C.-E., Shaw, F.-Z., Chen, Y.-H., Hsu, C.-H., and Chen, J.-Y. (2015). Combination of expert knowledge and a genetic fuzzy inference system for automatic sleep staging. *IEEE Transactions on Biomedical Engineering*, 63(10):2108–2118.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Magalang, U. J., Chen, N.-H., Cistulli, P. A., Fedson, A. C., Gíslason, T., Hillman, D., Penzel, T., Tamisier, R., Tufik, S., Phillips, G., et al. (2013). Agreement in the scoring of respiratory events and sleep among international sleep centers. *Sleep*, 36(4):591–596.

Mahowald, M. W. and Schenck, C. H. (2005). Insights from studying human sleep disorders. *Nature*, 437(7063):1279–1285.

Myllymaa, S., Muraja-Murro, A., Westeren-Punnonen, S., Hukkanen, T., Lappalainen, R., Mervaala, E., Töyräs, J., Sipilä, K., and Myllymaa, K. (2016). Assessment of the suitability of using a forehead EEG electrode set and chin EMG electrodes for sleep staging in polysomnography. *Journal of sleep research*, 25(6):636–645.

Najdi, S., Gharbali, A. A., and Fonseca, J. M. (2016). A comparison of feature ranking and rank aggregation techniques in automatic sleep stage classification based on polysomnographic signals. In *Bioinformatics and Biomedical Engineering: 4th International Conference, IWBBIO 2016, Granada, Spain, April 20-22, 2016, Proceedings 4*, pages 230–241. Springer.

Nitschke, J. B., Miller, G. A., and Cook, E. W. (1998). Digital filtering in EEG/ERP analysis: Some technical and empirical comparisons. *Behavior Research Methods, Instruments, & Computers*, 30:54–67.

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

O'Reilly, C., Gosselin, N., Carrier, J., and Nielsen, T. (2014). Montreal archive of sleep studies: an open-access resource for instrument benchmarking and exploratory research. *Journal of Sleep Research*, 23(6):628–635.

Peever, J. and Fuller, P. M. (2017). The biology of REM sleep. *Current biology*, 27(22):R1237–R1248.

Phan, H., Chén, O. Y., Tran, M. C., Koch, P., Mertins, A., and De Vos, M. (2021). Xsleepnet: Multi-view sequential model for automatic sleep staging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5903–5915.

Phan, H., Mikkelsen, K., Chén, O. Y., Koch, P., Mertins, A., and De Vos, M. (2022). Sleeptransformer: Automatic sleep staging with interpretability and uncertainty quantification. *IEEE Transactions on Biomedical Engineering*, 69(8):2456–2467.

Popovic, D., Khoo, M., and Westbrook, P. (2014). Automatic scoring of sleep stages and cortical arousals using two electrodes on the forehead: validation in healthy adults. *Journal of sleep research*, 23(2):211–221.

Rechtschaffen, A. (1968). A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects. *Public Health Service*, pages 1–55.

Ribeiro, A. H., Tiels, K., Aguirre, L. A., and Schön, T. (2020). Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness. In *International conference on artificial intelligence and statistics*, pages 2370–2380. PMLR.

Rodríguez-Sotelo, J. L., Osorio-Forero, A., Jiménez-Rodríguez, A., Cuesta-Frau, D., Cirugeda-Roldán, E., and Peluffo, D. (2014). Automatic sleep stages classification using EEG entropy features and unsupervised pattern analysis techniques. *Entropy*, 16(12):6573–6589.

Rojas, G. M., Alvarez, C., Montoya, C. E., De la Iglesia-Vaya, M., Cisternas, J. E., and Gálvez, M. (2018). Study of resting-state functional connectivity networks using EEG electrodes position as seed. *Frontiers in neuroscience*, 12:235.

Ronzhina, M., Janoušek, O., Kolářová, J., Nováková, M., Honzík, P., and Provazník, I. (2012). Sleep scoring using artificial neural networks. *Sleep medicine reviews*, 16(3):251–263.

Rusydi, M. I., Okamoto, T., Ito, S., and Sasaki, M. (2014). Rotation matrix to operate a robot manipulator for 2d analog tracking objects using electrooculography. *Robotics*, 3(3):289–309.

Schaltenbrand, N., Lengelle, R., Toussaint, M., Luthringer, R., Carelli, G., Jacqmin, A., Lainey, E., Muzet, A., and Macher, J.-P. (1996). Sleep stage scoring using the neural network model: comparison between visual and automatic analysis in normal subjects and patients. *Sleep*, 19(1):26–35.

Shambroom, J. R., Fábregas, S. E., and Johnstone, J. (2012). Validation of an automated wireless system to monitor sleep in healthy adults. *Journal of sleep research*, 21(2):221–230.

Shlens, J. (2007). Notes on kullback-leibler divergence and likelihood theory. *Systems Neurobiology Laboratory*, 92037:1–4.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE.

Sousa, T., Cruz, A., Khalighi, S., Pires, G., and Nunes, U. (2015). A two-step automatic sleep stage classification method with dubious range detection. *Computers in biology and medicine*, 59:42–53.

Stephansen, J. B., Olesen, A. N., Olsen, M., Ambati, A., Leary, E. B., Moore, H. E., Carrillo, O., Lin, L., Han, F., Yan, H., et al. (2018). Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy. *Nature communications*, 9(1):1–15.

Sun, C., Chen, C., Li, W., Fan, J., and Chen, W. (2019). A hierarchical neural network for sleep stage classification based on comprehensive feature learning and multi-flow sequence learning. *IEEE journal of biomedical and health informatics*, 24(5):1351–1366.

Supratak, A., Dong, H., Wu, C., and Guo, Y. (2017). Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel EEG. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008.

Thorpy, M. J. (2012). Classification of sleep disorders. *Neurotherapeutics*, 9(4):687–701.

Tsinalis, O., Matthews, P., and Guo, Y. (2015). Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders. *Annals of biomedical engineering*, 44.

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*, 44:1587–1597.

Van Sweden, B., Kemp, B., Kamphuisen, H., and Van der Velde, E. (1990). Alternative electrode placement in (automatic) sleep scoring (fpz-cz/pz-oz versus c4-at). *Sleep*, 13(3):279–283.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Y.-L. (2015). A wave energy converter with magnetic gear. *Ocean Engineering*, 101:101–108.

Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495.

Yao, D., Qin, Y., Hu, S., Dong, L., Bringas Vega, M. L., and Valdés Sosa, P. A. (2019). Which reference should we use for EEG and ERP practice? *Brain topography*, 32:530–549.

Zhang, S., McCane, B., Neo, P. S., Shadli, S. M., and McNaughton, N. (2020). Trait depressivity prediction with EEG signals via lsboost. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Zhang, X., Chen, F., and Huang, R. (2018). A combination of RNN and CNN for attention-based relation classification. *Procedia computer science*, 131:911–917.

Zhang, Y., Zhang, X., Liu, W., Luo, Y., Yu, E., Zou, K., and Liu, X. (2014). Automatic sleep staging using multi-dimensional feature extraction and multi-kernel fuzzy support vector machine. *Journal of healthcare engineering*, 5(4):505–520.

Zhou, Y., He, S., Huang, Q., and Li, Y. (2020). A hybrid asynchronous brain-computer interface combining ssvep and EOG signals. *IEEE Transactions on Biomedical Engineering*, 67(10):2881–2892.