Analyzing the Usefulness of a Low-Cost Respiration Sensor for Sleep Apnea Detection in a Clinical Setting

A Metric Based Approach

Morten Hamborg Andersen

Thesis submitted for the degree of
Master in Informatics: Programming and Networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2020

Analyzing the Usefulness of a Low-Cost Respiration Sensor for Sleep Apnea Detection in a Clinical Setting

A Metric Based Approach

Morten Hamborg Andersen

Abstract

Obstructive sleep apnea (OSA) is a common, but severely under-diagnosed sleep disorder characterized by repeated periods of reduced or paused breathing during sleep. *Polysomnography* (PSG) is the gold standard for the diagnosis of OSA, but requires overnight monitoring in a sleep laboratory, which is both resource-demanding and uncomfortable for patients. The main objective of the CESAR project is to increase the percentage of detected/diagnosed OSA patients with the use of cheaper sleep monitor solutions that can be applied at home. This should be achieved by utilizing low-cost consumer electronics instead. Which in turn should reduce the time and resources needed, as no PSG in a sleep laboratory is necessary for the initial diagnostic step. The idea is to use machine learning for automatic classification of OSA to eliminate the need for sleep experts in the first step.

In this thesis, we evaluate the usefulness of a low-cost strain-gauge respiratory effort sensor (FLOW) from SweetZpot for overnight monitoring of sleep apnea in a home respiratory polygraphy study. During the A3 study from Oslo University Hospital, we collect 57 FLOW and NOX recordings from 34 patients diagnosed with atrial fibrillation. We measure the signal quality produced by FLOW against the *respiratory inductance plethysmography* (RIP) NOX T3 sensor from NOX Medical by evaluating the classification performance of several machine learning models. This process requires the signal data from both sensors to be synchronized to utilize annotated scoring from a sleep expert. Our analysis reveals several data quality issues with FLOW related to connection loss, unreliable timestamping- and sampling rate, which proves to be a non-trivial problem to correct for synchronization. We discuss several approaches for timestamp adjustment and finally design a flexible window model that can both identify connection loss and adjust timestamps. We evaluate the FLOW recordings on a window-based approach to validate the timestamp adjustment (synchronization) and to analyze how the signal quality changes overnight. The signal quality evaluation is based on the breath detection accuracy metrics *sensitivity*, *positive predictive value* (PPV), and *clean minute proportion* (CMP), along with the breath amplitude accuracy metric *weighted absolute percentage error* (WAPE). These metrics are similar to how apnea and hypopnea events are scored by medical personal.

We achieve a sensitivity, PPV, CMP, and WAPE score of 97.2%, 94.2%, 59.4%, and 18.4%, respectively, indicating that our preprocessing is sufficient for mitigating the original data quality issues. Our results show that common behaviors during sleep, such as movement and changes in sleeping position, significantly affect the signal quality produced by FLOW, which we attribute to belt entrapment or misplacement. We are able to significantly increase the machine learning classification performance on FLOW data by applying a simple standardization. Using ten-fold-cross-validation, we achieve a classification accuracy of 76.1% using convolutional neural network. Our improvements suggest that preprocessing of the data results in better classification accuracy. For comparison, we achieve a classification accuracy of 79.6% on NOX.

Acknowledgments

First and foremost, I would like to thank my supervisors, Professor Dr. Vera Goebel and Professor Dr. Thomas Plagemann, for their guidance and support during the work of this thesis. Their encouragement and dedication are highly appreciated. I would also like to show my gratitude to researcher Stein Kristiansen for his input and assistance with the more technical aspects in this thesis. Finally, I would like to thank my partner of almost seven years, Kristin Marie, for her incredible support and patience.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Background and Motivation

Sleep Apnea (SA) is a common sleep disorder characterized by repeated periods of reduced or paused breathing during sleep. During disrupted breathing, the amount of oxygen (oxygen saturation) in the blood decreases. If the amount of oxygen in the blood becomes too low, the brain forces an awakening to resume breathing. As this disorder shows itself during sleep and the continuous awakenings are brief, sufferers are unlikely to remember it the next day. The repeated awakenings hinder deep sleep that consequently leads to daytime symptoms such as excessive sleepiness and feeling fatigued. SA is associated with serious diseases such as stroke, heart disease, diabetes, high blood pressure, anxiety, and depression. It can cause severe health implications for the sufferer, and, in the worst case, even death if left untreated [Young et al. 2002, 2004], [Punjabi 2008], and [Huang et al. 2008].

Obstructive sleep apnea (OSA) is the most common type of SA, which is characterized by recurrent episodes of partial or complete collapse of the upper airway during sleep. According to Hrubos-Strøm et al. [Hrubos-Strøm et al. 2011], one out of four middle-aged Norwegians are at high risk of having OSA, yet approximately 70-80% of all cases are expected to be undiagnosed [Punjabi 2008]. A study by McNicholas [McNicholas 2013] shows that SA sufferers have about two or three times increased risk of being involved in traffic accidents due to severe sleep deprivation. With numbers as high as these, it is apparent that proper sleep is vital for maintaining physical and mental health.

Polysomnography (PSG) is the gold standard and traditional procedure for diagnosing OSA, which requires the subject to spend the night in a sleep laboratory with several physiological sensors attached to the body. This process is very resource-demanding as it requires both expensive medical equipment, a laboratory, and trained medical personnel for supervision and evaluation of the results. Limited availability makes it impossible for doctors to prescribe polysomnography for everybody at risk of having OSA.

Besides, this type of sleep study can feel uncomfortable for many people as they have to sleep in an unfamiliar setting while being monitored, making it more difficult to fall asleep. As a consequence, a potential sufferer is less likely to seek medical help for a diagnosis.

Portable monitoring devices have been developed, as an alternative, in order to monitor sleep at home without the need for assistance. However, the recorded data still needs to be analyzed by sleep experts before a possible diagnosis can be determined. Besides, the equipment is usually still too expensive for the average person to buy as the number of sensors attached to the body is not significantly reduced compared to PSG.

The main objective of the CESAR project is to increase the percentage of detected/diagnosed OSA patients using cheaper polygraphy solutions that can be applied at home. Another goal of the CESAR project is to reduce the cost of polygraphy by utilizing low-cost consumer electronics. The idea is to use NOX T3 (medical certified polygraphy system) to compare the signal quality of cheaper sensors and evaluate their suitability for monitoring of OSA. Moreover, the CESAR project aims to reduce the time and required resources for diagnostic. Since PSG in a sleep laboratory is not necessary for an initial test, the idea is to apply machine learning techniques on the recorded sleep data for automatic classification of OSA to eliminate the need for sleep experts. The classification results should provide the user with a potential recommendation to visit a physician and help physicians with deciding whether polysomnography should be performed.

To reach this objective, it is fundamental that the cost of sensors is affordable for consumers and that they support different types of smartphones. Further, the signal of the sensors should be of good enough quality for correct classification. A study by Kristiansen et al. [Kristiansen et al. 2018], evaluates several machine learning algorithms based on their classification performance for detection of disrupted breathing using physiological data from two databases of different quality. The classification accuracy of the machine learning algorithms for all signal combinations is in the range of 90.6%-96.6% for the Apnea-ECG database and 58.2%-73.1% for the MIT-BIH database. The difference in accuracy clearly illustrates the importance of good data quality for the detection of disrupted breathing.

1.2 Problem Statement

To conduct an initial SA test at home, cheaper consumer grade sensors are an affordable alternative to certified medical grade equipment. The general assumption that quality is related to price, does not necessarily mean that these sensors cannot be used for an initial SA test. In recent studies by Løberg [Løberg 2018; Løberg et al. 2018], the signal quality of FLOW, a low-cost respiratory effort sensor, have shown promising results under controlled lab conditions. However, the question remains unanswered whether the sensor is useful for unattended overnight sleep monitoring at home. In this context, the *usefulness* of the sensor is reflected in the classification performance of machine learning models that can be achieved with this sleep monitoring data. Therefore, the overall problem statement can be summarized in one question:

- Is the classification performance on data from FLOW sensors during unattended overnight sleep monitoring at home good enough?

To answer this question, we analyze how accurate the FLOW sensor is for SA detection through empirical studies based on: recordings from real patients, existing metrics suggested in previous work, and by testing it on machine learning classifiers. As such, the overall problem statement breaks down into more specific parts, which we address with the following questions:

- How good is the quality of the collected data?
- How good and consistent is the signal quality from FLOW sensors during unattended overnight sleep monitoring at home?
- How good is the performance of machine learning classifiers on the collected data?
- How can preprocessing increase the performance of classifiers?

1.3 Outline

The structure of this Master Thesis is as follows:

- **Chapter 2 – Background**
  In this chapter, we present an overview of sleep apnea, along with the contributions from the CESAR project and an introduction to machine learning.
- **Chapter 3 – Data Quality Issues**
  This chapter presents data quality issues in the dataset from the FLOW sensors and the difficulties with measuring the quality of data accurately. Furthermore, we describe our approach to evaluate the quality of this dataset.
- **Chapter 4 – Preprocessing and Implementation**
  In this chapter, we discuss how to correct the data quality issues in data from the FLOW sensor, with the design of a flexible window model as our solution. Furthermore, we present our Python implementation of the steps for preprocessing the dataset.
- **Chapter 5 – Evaluation**
  In this chapter, we evaluate the result of our preprocessing and the quality of overnight recordings. Furthermore, we compare and evaluate the classification performance of several machine learning classifiers based on their ability to detect disrupted breathing in signal data from FLOW and NOX with a comparison with related work.

- **Chapter 6 – Conclusion**
  Finally, this chapter concludes our findings with a critical assessment of this work and a discussion of future work.

Chapter 2

Background

This chapter presents an overview of sleep apnea, along with the contributions from the CESAR project, and an introduction to machine learning. We begin by describing the characteristics, symptoms, risks, and diagnostic challenges of sleep apnea in Section 2.1, with an emphasis on the most prevalent kind, i.e., obstructive sleep apnea. We continue in Section 2.2, by describing the plan and objective of the CESAR project along with the current status and contributions. The most important elements of machine learning and techniques are presented in Section 2.3. Finally, we conclude this chapter in Section 2.4.

2.1 Sleep Apnea

SA is a common sleep disorder characterized by repeated periods of reduced or paused breathing during sleep [Punjabi 2008]. During disrupted breathing, the amount of oxygen (oxygen saturation) in the blood decreases. If the amount of oxygen in the blood becomes too low, the brain forces an awakening to resume breathing. As this disorder shows itself during sleep and the continuous awakenings are brief, sufferers are unlikely to remember it the next day. The repeated awakenings hinder deep sleep that consequently leads to daytime symptoms such as excessive sleepiness and feeling fatigued. The lack of any recollection of nightly awakenings means that people can easily misinterpret these symptoms for something else. The most notable effect of the disorder is a lower quality of life, though it can lead to more serious complications if it remains undiagnosed. SA is associated with a higher risk of having diabetes, heart disease, stroke, depression, anxiety, high blood pressure, and motor vehicle accidents [Young et al. 2002, 2004], [Punjabi 2008], and [Huang et al. 2008]. The severe consequences show the importance of proper sleep patterns to maintain both physical and mental health throughout life.

SA is one of the most treated sleep disorders in sleep clinics amongst insomnia, restless leg syndrome, and narcolepsy [Alaska Sleep Clinic 2018b]. However, approximately 70-80% of all SA cases remain undiagnosed [Punjabi 2008]. It is estimated that 25% of the Norwegian middle-aged population is at high risk of suffering from obstructive sleep apnea [Hrubos-Strøm et al. 2011]. OSA sufferers are about two or three times as likely to be involved in traffic accidents because of severe sleep deprivation [McNicholas 2013] and [Gottlieb et al. 2018]. Early diagnosis of SA is vital for reversing or alleviating the disorder. To summarize; if SA remains untreated, it is clear that it has a severe impact on the health of individuals, as well as society as a whole.

There are two main types of sleep apnea, namely *Obstructive Sleep Apnea* (OSA) and *Central Sleep Apnea* (CSA), in addition to a more recently discovered type, which is a combination of the two - known as *Mixed* or *Complex Sleep Apnea*. Studies have shown that OSA accounts for most cases of SA and that people start to exhibit symptoms of CSA when treated for OSA, indicating that they suffer from complex sleep apnea [Morgenthaler et al. 2006]. In this work, we focus on OSA since it is the most prevalent type of SA.

2.1.1 Obstructive Sleep Apnea

Obstructive Sleep Apnea (OSA) is characterized by recurrent episodes of partial or complete collapse of the upper airway during sleep [ASAA 2020a]. If the muscles that control the tongue and the soft palate relax too much during sleep, both can fall back and cause the upper airway to become narrower and constricted. An illustration of regular airflow is shown in Figure 2.1a, while in Figure 2.1b and Figure 2.1c, the tongue falls back, which partially or completely blocks the airways. A complete blockage inhibits breathing, while a partial blockage will make the sufferer breath more shallow. In both cases, the sufferer will automatically increase the respiratory effort by pressing more air through the blockage in order to resume breathing. For this reason, loud snoring is often associated with OSA, along with gasping during the brief awakenings.

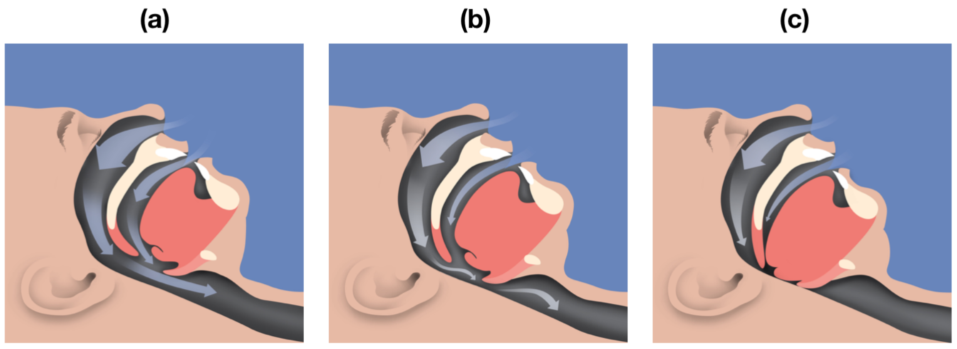

Figure 2.1: Illustration of physical blockage of upper airways [SomnoMed 2020]

2.1.2 Symptoms

A significant challenge with OSA is that it occurs when people are sleeping, thus it is hard to notice. A contributing factor is that the most common symptom is daytime sleepiness. Feeling tired can be normal for many people due to various reasons, and since they are not aware of the nightly awakenings, the disorder is very likely to go unnoticed. The obstruction of the airways often results in loud snoring, which means that bedside partners or family members in the house are often the first to raise suspicion of someone having SA. Other symptoms include *frequent breaks in breathing, restless sleep, headaches, mood swings, and lack of concentration* [Alaska Sleep Clinic 2018a].

Risk Factors

Several factors increase the risk of someone having OSA. The condition is directly linked to weight as obese people have an increased amount of fatty tissue that builds up in the neck and throat, which can lead to narrowing of the airways [Alaska Sleep Clinic 2018a]. Other factors are age and sex. Middle-age males have the highest risk due to aging-related weakening of muscle tone, lengthening of the soft palate, and increased deposition of fat in the throat [Punjabi 2008]. Lifestyle habits like smoking can cause inflammation and fluid retention in the upper airways, which can impede the airflow. Frequent alcohol use is another factor, as alcohol can cause the throat muscles to relax to such an extent that the airways become blocked.

Treatment

Common treatments of OSA include changes in lifestyle, such as quitting smoking and losing weight. Although OSA can affect anyone, it is more prevalent in people that sleep on their backs. Therefore, another option is to use a special device to prevent people from sleeping on their backs. In more severe cases of OSA, doctors often recommend *continuous positive airway pressure* (CPAP), which is a face mask device that pressurizes the upper airways of a person to keep them open during sleep [Punjabi 2008].

2.1.3 Diagnostic

To diagnose OSA, potential sufferers need to undergo a sleep study to capture the nocturnal events (i.e., nightly awakenings) as they are happening at night using various types of physiological sensors. The severity of OSA is commonly determined based on the average number of *apnea* and *hypopnea* events per hour of sleep, referred to as the Apnea-Hypopnea Index (AHI). According to the American Academy of Sleep Medicine (AASM) [Berry et al. 2012], this metric is mandatory when diagnosing OSA. AHI is divided into four severity classes: normal sleep, mild OSA, moderate OSA, and severe OSA, which are classified as follows [WebMD 2020]:

- Normal sleep: An AHI of fewer than 5 apnea/hypopnea events, on average, per hour

- Mild OSA: An AHI of 5 to 14 apnea/hypopnea events per hour
- Moderate OSA: An AHI of 15 to 29 apnea/hypopnea events per hour
- Severe OSA: An AHI of 30 or more apnea/hypopnea events per hour

Breathing disruptions are, in general, classified as either an *apnea*, or a *hypopnea*. The difference between the two is that *apnea* refers to the complete cessation of breathing, while a *hypopnea* refers to a reduction in airflow or shallow breathing. According to the scoring rules outlined by AASM [Berry et al. 2012], a breathing stop has to last for at least ten seconds to be classified as an apnea. Hypopnea is classified when there has been a reduction in breathing of at least 30% that lasts a minimum of ten seconds, in addition to detecting either a $\geq$3% drop in oxygen saturation. Arousal means that the person has woken up for three to fourteen seconds, which can be detected using the EEG signal.

Polysomnography

Polysomnography (PSG) is the most common method used for objective assessment of sleep and is the gold standard for the diagnosis of OSA. In a clinical setting, it requires overnight monitoring in a sleep laboratory with several physiological sensors attached to the patient's body (see Figure 2.2). There are several benefits of PSG stemming from its long documented history use in the field. The equipment is often very expensive and of good quality, formally tested and medically certified, thus generally produce data of high quality and precision. Additionally, during PSG, it is required that trained sleep technologists monitor the subject, making sure that all sensors are properly attached. Using their experience in analyzing sleep results, they can determine if the recorded data is enough for a diagnosis or if another sleep session is required. They may also ask the subject to change the sleeping position to try and sleep on their back since OSA is often most prevalent in this position.

There are also some drawbacks to PSG. First of all, it is very resource-demanding as it requires medical trained personal to monitor the subject during the night and to analyze the recorded sleep data afterward. The excellent quality of PSG comes at the cost of limited availability, because of expensive equipment and the need for trained sleep technologists. Besides, the subject has to sleep in an unfamiliar environment with many sensors attached, which can make them feel uneasy and stressed, possibly hindering any sleep at all. As a result, there is often a waiting time for PSG, and practitioners cannot recommend everybody at risk of having OSA for PSG. As a consequence, the threshold for people to seek medical help may be high, along with an unnecessarily long diagnostic time starting from the initial suspicion of OSA to a medical diagnose.

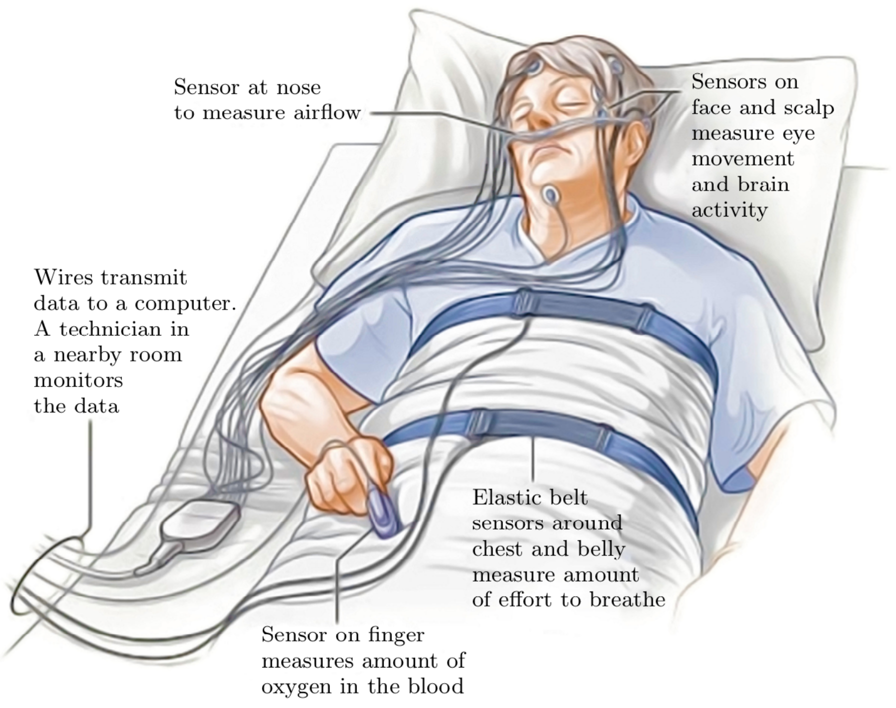

Figure 2.2: Illustration of traditional polysomnography [Medifixit 2016]

Physiological Signals

The variety of physiological sensors used during PSG largely contributes to its reliability for the detection of OSA. Traditional PSG must include at least the following signals: electroencephalography (EEG), electrooculography (EOG), electrocardiogram (ECG), chin electromyography (EMG), limb EMG, the respiratory effort from the thorax (chest) and abdomen, nasal airflow, and pulse oximetry. Common for EEG, ECG, EMG, and EOG are that they measure the electrical activity produced by the brain, heart, muscles, or eyes by attaching electrodes to the body.

PSG studies often includes a microphone, accelerometer, and body position sensors to capture and monitor snoring sounds and sleeping positions. PSG is used to determine a variety of sleep disorders besides OSA and SA in general, including narcolepsy, idiopathic hypersomnia, periodic limb movement disorder, Rapid Eye Movement (REM) sleep behavior disorder, and parasomnias. Continuing, we will only describe the most used sensors as not all sensors included during PSG are relevant for the detection of OSA.

Since respiration is at the core of OSA, signals that monitor the respiratory effort are crucial. They are divided into two types that either measure the respiratory effort or the airflow. The gold standard for measuring airflow is a *pneumotachograph*, which is a mask placed over the mouth and nose. The mask is, however, bulky and large enough to reduce sleep comfort and is therefore not used in most sleep studies. Instead, AASM recommends non-intrusive sensors such as a *nasal pressure transducer* or *oronasal thermal sensor* to measure airflow and for monitoring both apnea and hypopnea events [Berry et al. 2012].

Sensors that monitor the respiratory effort, instead of airflow, measure the actual physical effort of breathing. There exists a wide variety of non-invasive sensor types for such a purpose, but according to Berry et al. [Berry et al. 2012] only the *respiratory inductance plethysmography* (RIP) and *polyvinylidene fluoride* (PVDF) type sensors are recommended for SA monitoring. Both sensor types are placed on belts and strapped around the patient to measure the expansion and contraction of the abdomen or thorax associated with breathing. PVDF sensors are typically located in a single portion of the belt and depend on the remainder of the belt to transfer the force (due to stretching of the belt) to the sensor part. RIP belts, on the other hand, measure the inductance of the belt as a whole, which means they detect a change in length of any sections of the belt [Bronstein et al. 2017].

During the scoring of PSG data, sleep experts rely on the signal from respiratory effort sensors to distinguish between the two types of apneas, namely central and obstructive. The difference between the two types is that the sufferer will show no effort of breathing during a central apnea, which means the signal from respiratory effort sensors will flatline. In this case, the issue lies with the brain, failing to transmit the proper signals to the muscles that control the breathing [Mayo Clinic 2020].

EEG and EOG sensors attached on the face and scalp (see Figure 2.2) measures the brain activity and eye movement, respectively, during sleep. Both sensors are often used in sleep studies to determine wakefulness and the sleep stages. This is important for correctly calculating the AHI index since we want to exclude the period when the patient is falling asleep, and because arousals are associated with hypopnea events.

A pulse oximeter sensor is a small clip that is often attached to the finger (see Figure 2.2) to measure the oxygen saturation in the blood. The sensor is essential to classify hypopnea events, that is, to identify periods with an oxygen drop of $\geq$3%. The oxygen saturation derived from a pulse oximeter is referred to as SpO_2 and is an indirect measurement as it is often calculated over a *time span*, making it a *delayed* signal.

A microphone can be included during PSG to capture snoring sounds and classify OSA as snoring is a very common symptom. Body position sensors can be used to determine the sleeping position of the subject. In some cases, this may be important information as sleeping on the side rather than the back significantly lowers the number of apnea/hypopnea events for some people, which may need to be considered to avoid misclassification.

2.1.4 NOX T3 Sleep Monitor

Respiratory polygraphy is a cost-effective alternative to PSG that can be performed at home without the assistance of medical personnel or monitoring. Letting people use a portable monitoring device at home reduces the resources needed, but only to some extend. The recorded sleep data still needs to be manually analyzed by a sleep expert before an eventual diagnosis can be determined. There exists a variety of portable devices, with NOX T3 made by NOX Medical, often being the preferred choice for the diagnosis of sleep-related disorders in hospitals around the world. NOX T3 is a complete medical grade respiratory sleep monitor that comes equipped with all of the physiological sensors used in traditional PSG. The minimalistic design of NOX T3 makes it easy to use as it is battery-powered and comes with internal storage [NOX Medical 2020a].

The main drawback of portable respiratory polygraphy devices is the price. For instance, the cost of NOX T3 is estimated to be around *77 000 - 100 000 NOK*, making it far too expensive for personal consumers to buy for an initial diagnosis at home. Respiratory polygraphy can be considered as a portable version of traditional PSG to provide another option for the clinical diagnosis of OSA. A recent study by Xu et al. [Xu et al. 2017] validates the performance of NOX T3 for home sleep testing to diagnose OSA in adults. The clinical/medical certification of NOX T3 implies that it is priced towards hospitals and not for personal consumers. NOX T3 is used in the CESAR project for several purposes, such as to evaluate the performance of machine learning classifiers on physiological signals and to compare the signal quality against cheaper sensors like FLOW, as in our case. We also use the manual scorings done on NOX T3 data from a sleep expert later on. An illustration of NOX T3 is shown in Figure 2.3. The device used for recording and storing data is shown on the left side of the figure and is attached to the chest on top of the respiratory effort sensor that measures the thorax expansion, as shown on the right side of the figure.

FLOW Sensor

The FLOW sensor from SweetZpot is an affordable strain-gauge respiratory effort belt, priced at about *200 Euros* [SweetZpot 2020]. In addition to capturing the respiratory effort, the sensor can measure the heart rate when the belt is worn directly on the skin. The sensor uses batteries as power-source and transmits the data over Bluetooth to a smartphone using the Android application called *RawDataMonitor*. The FLOW sensor should capture the respiratory data at a sampling rate of 10 Hz. FLOW was originally developed for training/fitness purposes, such as monitoring the breathing of cyclists or rowers. Monitoring breathing in real-time can give athletes a better insight into how their body is performing and how their breathing changes during different levels of training intensity. This data can then be used by athletes to optimize their performance.

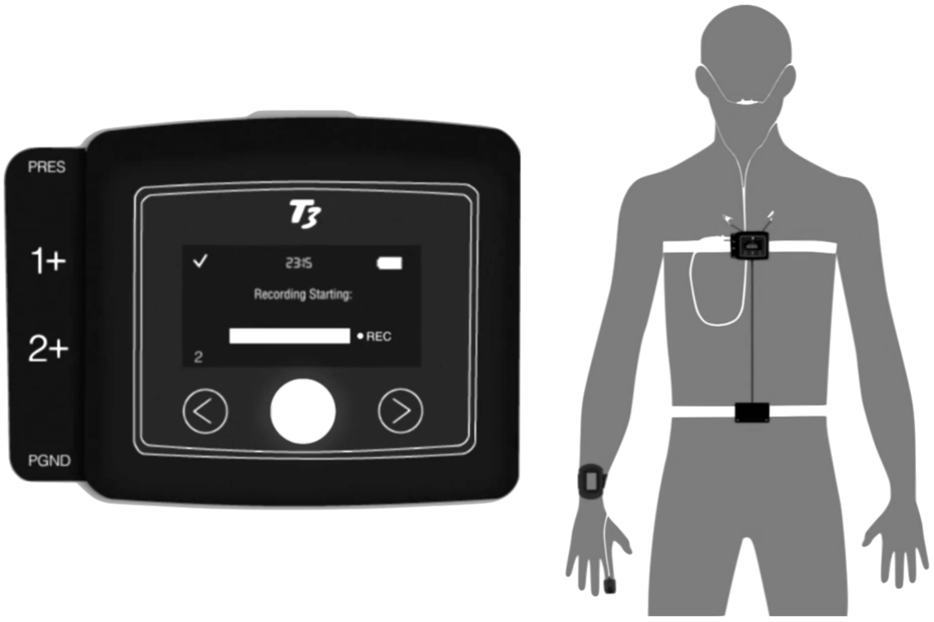

Figure 2.3: Illustration of NOX T3 [NOX Medical 2020a]

The FLOW sensor might be useful for SA monitoring since it captures the breathing effort. However, this context is different from the intended training contexts. For instance, a cyclist will usually attach the mobile device on the handlebar of the bike. As a result, the short distance between the sensor attached to the chest and the smartphone means the connection is free from obstacles. Furthermore, such activities usually do not last for more than a couple of hours. For SA monitoring, on the other hand, the sensor needs to record data for an entire night which is more demanding. Besides, if a person is sleeping on the sensor part of the belt, the connection may become blocked.

2.2 CESAR Project

The long-term goal of the CESAR project is to increase the percentage of diagnosed OSA cases and reduce the time it takes for diagnosis. In addition to support monitoring of patients in the long-term with cost-efficient tools that are user-friendly for sleep analysis at home. Instead of providing alternatives to traditional PSG diagnosis, CESAR aims to reduce the threshold for people to seek medical help by enabling them to take the first step towards an early diagnosis at home. The core idea is that people should be able to buy an affordable sensor, such as a respiratory effort belt or a smartwatch, for use together with a smartphone to record sleep data. The goal is to utilize machine learning techniques to automatically classify the recorded sleep data and provide the user with an indication of OSA severity and further recommendation.

For this to be realistic, the signal data produced by these sensors need to be of adequate quality, while at the same time being affordable for consumers. Suitable machine learning classifiers should be determined. These should have a high performance on physiological time-series signals that can learn the relevant patterns for OSA.

2.2.1 Data Mining for Patient Friendly Apnea Detection

One of the first contributions of the CESAR project is a study [Kristiansen et al. 2018] that analyzes whether a subset of the physiological signals used in traditional PSG allows for detection of SA events using automatic classification. In this study, the signals in focus are the oxygen saturation measured with a pulse oximeter, in addition to respiration from the abdomen, chest, and nose. The performance of five data mining techniques in relation to SA classification are evaluated using two datasets from PhysioNet. The two datasets contain PSG data from several healthy subjects and SA sufferers. An accuracy of 96.6% is achieved using a combination of respiration data from the chest and nose. This highlights the ability to perform automatic scoring to reach similar classification as trained medical personal. Their findings clearly illustrate the importance of data quality as the accuracy of the data mining classifiers on all signal combinations, is in the range of 90.6%-96.6% for the Apnea-ECG dataset and 58.2%-73.1% for the MIT-BIH dataset. Furthermore, this study suggests that one signal might be sufficient for the detection of disrupted breathing if the data size and quality are good enough. Finally, this study concludes that the respiratory effort from the abdomen is the preferable signal choice when considering both the performance of classification and patient comfort.

2.2.2 Signal Quality Evaluation of Respiratory Effort Sensors

Based on the prior result of the respiratory effort sensors for automatic detection of SA, a study by Løberg [Løberg 2018] compares the signal quality of several low-cost respiratory effort sensors with emphasis on SA monitoring. The best way to accurately determine the quality of respiratory effort sensors is to include them during a sleep study and let sleep experts manual score the data afterward. However, this is very resource-demanding and not feasible, considering that multiple sensors should be evaluated. Instead, Løberg determines the quality of sensors by capturing the respiratory effort of several participants in a laboratory while awake. The participants performed several tasks to simulate sleep apnea events by, for example, holding their breath or breathing shallow for an extended period of time. Only shorter sessions were conducted because the subjects needed to be awake the whole time.

A RIP sensor from NOX Medical is used as the gold standard to compare the different respiratory effort sensors against. Løberg defines several suitable metrics for measuring the quality of respiratory effort sensors closely related to how medical personnel score apnea and hypopnea events. The *sensitiv-*

ity, *positive predictive value*, *clean minute proportion* are used to measure and evaluate the breath detection accuracy of the signal produced by respiratory effort sensors. In addition, the *weighted average percent error* metric is used to measure and evaluate the breath amplitude accuracy.

The signal from FLOW achieves the best breath amplitude accuracy of the sensors evaluated. A reason for this might be because it is the only belt type sensor that captures the same unit of measurement as NOX, which is *lung volume* (tidal volume). As a result, the CESAR project aims to use the FLOW sensor in future studies.

2.2.3 A3 Study

Including a sensor in a clinical setting is usually an extensive process that requires planning, medical equipment, personal and patient recruitment. The A3 study from Oslo University Hospital presents us with a unique opportunity because of the collaboration with the CESAR project. Besides, the proximity between the Institute for Informatics and the Oslo University reduces the time spent with collecting the FLOW sensors from patients for data transfer and cleaning before further use. During the fall of 2018, the remaining patients in the A3 study yet to undergo home polygraphy, were asked and agreed to use the FLOW belt in addition to the NOX T3 sleep monitor during sleep.

Since the quality of FLOW showed promising results during the initial study by Løberg with regards to measuring the respiratory effort, we decided to use the FLOW sensor with the last 34 patients in the A3 study during the fall of 2018. These patients used both the NOX T3 Sleep Monitor and the FLOW belt during their participation because the CESAR project wants to study how low-cost sensors for specific respiratory signals compare to certified medical-grade sensors for SA monitoring. The polygraphy data from the NOX T3 is manually scored by a sleep expert, which is very beneficial for us because it enables us to use the scoring of NOX T3 recordings to analyze FLOW data (to determine SA events). The scoring results by a sleep expert would likely be less accurate using the signal from FLOW alone. The reason is that not all SA related events are easy to distinguish when using only one respiratory signal. Based on how hypopnea events should be scored according to AASM, accurate scoring requires monitoring of the oxygen saturation level and detection of arousals. However, it may be possible that machine learning algorithms can learn to detect hypopnea events from only one respiratory signal better than experts.

The motivation behind the A3 study is to investigate the prevalence, risk factors, characteristics, and type of SA in ablation candidates with paroxysmal atrial fibrillation (AF) [Traaen et al. 2019]. The main findings from the A3 study are that SA is common in this AF population, with OSA being the most prevalent type. The results indicate that SA is under-diagnosed in patients suffering from AF.

To summarize, we have collected recordings from 34 patients during the last part of the A3 study in the fall of 2018. Our dataset consists of patients that have not been diagnosed with OSA nor receives OSA treatment. None of the patients have been diagnosed with persistent or no AF in the last three months. Our population of 34 patients underwent two polygraphy sleep tests at home using both NOX T3 and FLOW during their participation. The dataset should consist of 69 nights of recording since one sleep test was redone because of poor data quality. For various reasons, some recordings are missing or excluded, which brings the dataset from 69 down to 57 nights of recording. Some nights the FLOW sensor has only recorded for a small period of the night. For this reason, some recordings are excluded because they are too short to be used for evaluating the quality of FLOW.

2.3 Machine Learning

We use supervised learning algorithms to evaluate NOX T3 and FLOW data. These algorithms rely on a training dataset that consists of a set of examples for *learning*. By building a mathematical model based on the inputs and the desired outputs from the training data, the goal is that these algorithms can accurately predict the outcome from similar data in the future [Alpaydin 2014]. In our context, the signal data from either FLOW or NOX is the input, while the desired output is the corresponding manual scoring by a sleep expert. By training an algorithm only on the signal from FLOW, we can compare the prediction accuracy to the prediction accuracy of the same algorithm trained on the signal from NOX.

Since we are not interested in predicting the type of SA event, all SA related events are combined into one single class of disrupted breathing. As a result, the classification is a two-class problem, which means that an epoch is either classified as disrupted breathing or normal breathing. The epoch size determines the maximum number of SA events that a classifier can detect per hour. A two-minute epoch size means that a maximum of 30 SA events per hour, although there might be more than one SA event in each epoch. The estimated AHI index may be lower than expected as a result. It is also possible to overestimate the AHI index by using a small epoch size. For instance, some breathing stops may last for thirty seconds or more, which means that the same SA event can span over subsequent epochs if using a thirty-second epoch size or smaller. We have chosen to use a one-minute epoch during our ML evaluation, which means that 60 SA events are possible per hour. This is large enough to detect severe SA, i.e., AHI index of 30 or more.

Some of the standard metrics for evaluating the classification of a two-class problem are sensitivity, specificity, and accuracy [Wikipedia 2020j]. For a two-class problem, we want the algorithm to learn to identify the so-called target class. For SA detection, disrupted breathing is the target class we want the algorithm to learn to detect.

The sensitivity metric determines the algorithm's ability to detect disrupted breathing, while the specificity metric determines the algorithm's ability to detect normal breathing. The accuracy metric combines sensitivity and specificity to describe the proportion of epochs that have been correctly identified. The performance of several algorithms is often compared using the accuracy metric. In some cases, however, a high sensitivity score might be favored at the cost of a low specificity. One example is a metal detector at an airport used to dangerous objects (i.g., weapons), that also detect belts and watches that people wear. Therefore, the kappa statistic is commonly used to indicate the degree of agreement by different metrics, which is useful for comparing algorithms. A kappa value of 1 means there is a perfect agreement between the metrics, while a kappa value of 0 implies that there is no effective agreement between the metrics. Finally, a negative kappa value means the agreement is worse than random [Wikipedia 2020g].

2.3.1 Approach

Choosing the best machine learning technique depends on the context in question and the goal of the classification. Another aspect to consider is the time and computation power needed for training a classifier. We have chosen to use primarily deep learning algorithms. The selection is based on performance, computational efficiency, and the fact that these algorithms are already implemented in the CESAR project. Deep learning algorithms are inspired by the human brain in the sense that they *learn* from experience similarly by repeatedly performing a task to improve the outcome [Wikipedia 2020i]. The classifiers, except for random forest, are split between feedforward and recurrent neural networks. The main difference between the two types is that feedforward networks only allow information to move forward, while recurrent networks allow information to be reused as input.

Multilayer Perceptron

Multilayer Perceptron (MLP) is a feedforward artificial neural network that consists of at least three layers of nodes, namely an input layer, a hidden layer, and an output layer. However, MLPs do often consist of multiple hidden layers. MLPs are fully connected, which means that each node in one layer connects with a certain weight to every node in the following layer [Wikipedia 2020b]. For this reason, MLPs are prone to overfitting, meaning they learn the training data too well, which reduces their ability to generalize future data.

Convolutional Neural Network

Convolutional Neural Network (CNN) is another feedforward neural network. CNN is one of the most popular deep learning models, most commonly applied to image classification. Over the last few years, CNN has shown surprisingly promising results in other fields, such as in recommender systems and natural language processing. Besides the high accuracy in image classification, a key factor for its popularity is the computational efficiency, enabling the possibility to run the model on smartphones. Briefly explained, CNN can be viewed as a regularized version of MLPs that reduces the risk of overfitting. CNN takes advantage of hierarchical patterns in the data to assemble complex patterns using more simple patterns. CNN requires little preprocessing as it learns the filters and features which are important for the specific classification context [Wikipedia 2020c].

Long Short-Term Memory

Long Short-Term Memory (LSTM) is a recurrent neural network with feedback connections, well-suited for classification on time series data where there often can be a large time difference between subsequent events of interest. LSTM is typically composed of a cell and three gates; an input, output, and forget gate [Wikipedia 2020d]. The gating mechanism regulates the flow of information in and out of the cell, thus the cell can remember values over arbitrary time intervals. In addition to LSTM, we use some variations of the algorithm. In bidirectional LSTM (BILSTM), time-series data can propagate backward in time, in addition to forward. In stacked bidirectional LSTM (SBILSTM), there is not only one, but multiple LSTM layers stacked on top of each other. The deeper learning should, hopefully, increase the accuracy, at the cost of more complexity. The BIWALSTM is a bidirectional LSTM that that provides LSTM with an attention property suggested by [Wang et al. 2016]. This should increase performance and possibly give insight into which part of an epoch the algorithm uses to determine whether the epoch contains disrupted breathing or not.

Gated Recurrent Units

Gated Recurrent Units (GRUs) is another gating mechanism like LSTM, where the main difference is that GRUs lacks an output gate [Wikipedia 2020e]. Although GRUs are less sophisticated than LSTM, they have proven to be as good or better than LSTM with a smaller dataset.

Random Forest

A random forest (RF) classifier is constructed by multiple decision trees, where each tree is trained on a slightly different subset of the data [Wikipedia 2020a]. The prediction of all trees are averaged and serves as the final prediction of the RF. The main point of a RF classifier is that it tries to correct for decision tree's tendency of overfitting and memorizing their training set.

2.3.2 k-fold Cross Validation

K-fold cross-validation is a popular technique used to evaluate machine learning models by randomly splitting the data into *k* equal sizes, using *k-1* subsets to train the model and the remaining subset for testing and evaluation [Wikipedia 2020f]. The key point of this approach is that cross-validation is repeated *k* times (the folds) and that each subset is used exactly once for testing and validation and *k-1* times for training. The result of each fold is then averaged to produce a final estimation of the model, including the standard deviation of each metric. One advantage of cross-validation is that we can inspect the result of each fold for obvious issues. For example, if the accuracy in one fold is much lower than the remaining folds, likely, one or more of the recordings used for testing and validation in this particular fold has more variation in the data compared to the remaining recordings. In general, a higher number of folds should reduce the significance of error from bias, but the same time increases the errors related to the higher variance of data. Another relevant factor to consider is the size of the dataset, as the computational time increases with each added fold.

2.4 Discussion and Conclusions

Sleep apnea is a severely under-diagnosed disorder that is hard to suspect because it shows itself during sleep. Traditional PSG sleep studies are very resource-demanding and uncomfortable to undergo. Besides, the threshold for people to seek medical help is often high. The CESAR project aims to lower this threshold by enabling people to take the first step towards a diagnosis at home using various types of low-cost sensors. There are several important physiological parameters to consider for SA monitoring which are: airflow, respiratory effort, wakefulness, and oxygen saturation.

FLOW is a low-cost respiratory effort sensor that the CESAR project has evaluated in a controlled environment. Based on the results, we include the sensor in a clinical sleep study, along with NOX T3 for comparison, to evaluate the usefulness of FLOW for SA detection. The evaluation is based on signal quality metrics and the classification performance of various ML models regarding the detection of disrupted breathing.

The data quality issues we discovered in the FLOW data need to be addressed to synchronize the signal with NOX for evaluation. As a result, we discuss how to preprocess the dataset in the following chapter.

Chapter 3

Data Quality Issues

In this chapter, we present an overview of data quality issues and how we can assess the quality of data, with an emphasis on respiratory effort sensors (like FLOW). We begin in Section 3.1 with a brief introduction to the factors of data quality, followed by a description of the data quality dimensions and metrics we use to measure the quality in Section 3.2. Next, we present a baseline analysis of the data quality issues for FLOW in Section 3.3. We finally conclude the chapter in Section 3.4.

3.1 Data Quality Assessment

The definition of *quality* is related to the wish of deciding how one object (or some *thing*) compares to similar objects and the performance of that object for a given task. Therefore, data quality can be defined as the data's purpose for a given context, which means the capability of data to satisfy the stated business, system, and technical requirements of an enterprise, that is, its *fitness for use* [Mahanti 2019]. In regards to our context, the quality of the collected data is, first and foremost, reflected in its usefulness for the detection of disrupted breathing.

The first dataset inspection and analysis is useful to get a first impression of the general quality of data in question. Tools that allow for a visual representation of data can often help uncover underlying issues. Knowledge of the data and context is fundamental to determine the right approach and steps for preparing and measuring the quality of the dataset. The aim is to assess data quality as accurately as possible to provide a foundation for further decision makings. The motivation behind can be complex and depends on the stated business. One reason is the wish to improve the quality of existing data. Using well-defined metrics that are easy to understand means that different attempts to improve the data quality can be compared to each other. Another reason is the wish to improve the process or equipment used to collect data. For example, if we want to improve the quality of the sensor, we can use metrics to measure the captured data from various versions of the sensor for comparison.

The uniqueness of our dataset means that we cannot redo the recordings because of any data issues we may discover. Therefore, our focus is to improve the quality of the obtained data if necessary. As a side effect, possible data quality issues we discover should be useful for improving the quality of future recordings.

3.1.1 Data Quality Dimensions

A Data Quality (DQ) dimension represents a particular feature of some data that can be measured to analyze and determine the quality of the data. In other words, each DQ dimension describes a way to view the data, which is useful to understand, analyze, and hopefully resolve or minimize data quality issues. Askham et al. [Askham et al. 2013] defines six generic key DQ dimensions with the hope that data quality practitioners adopt them as the standard method for assessing and describing the quality of data. The six dimensions are *completeness*, *uniqueness*, *timeliness*, *validity*, *accuracy*, and *consistency*. The motivation behind the study is to reduce confusion since practitioners have different views and use different terms to describe the same dimension. However, there is no definitive list of DQ dimensions that can be used for assessing the quality of data, with the confusion still being present. For example, Mahanti [Mahanti 2019] lists thirty different DQ dimensions, including both *accuracy* and *correctness*, which are often used interchangeably by practitioners even though these may be conceptually different. The main takeaway point is that not all DQ dimensions of data should or need to be measured, which is also infeasible for us, given the limited time frame. Therefore, our focus is to determine the DQ dimensions that are most important to consider when measuring the signal quality of recordings and the quality of the dataset as a whole.

3.1.2 Physiological Time Series

One of our main objectives is to measure the signal quality of overnight recordings to understand how reliable the FLOW sensor is. As previously explained in Section 2.2.2, one of the contributions of the CESAR project is a study by Løberg [Løberg 2018] which defines several metrics that are useful for measuring the quality of respiratory effort sensors, like FLOW. Based on Løbergs result, we decided to utilize the same metrics and scripts as much as possible.

Out of the six generic DQ dimensions, only the *accuracy* and *completeness* are used to measure the signal quality. This is partly because it is difficult to measure the *validity* of a signal and because the *uniqueness*, *timeliness*, and *consistency* are irrelevant when assessing the quality of individual time series. The two dimensions we use are described as follows:

Completeness: The completeness dimension describes the difference in the length of a recording compared to its intended length. For example, a person that monitors the respiratory effort overnight for eight hours, but the recording only contains four hours of data. In this case, the completeness of the data is 50%. Monitoring in an unattended setting means it is challenging to determine what the intended length is. As we are recording with two different device, one option is to use the length of the recording with the NOX T3 to measure the completeness of the FLOW recording. However, the length of the NOX T3 recording is not necessarily an accurate description of the intended length, because we are mainly interested in the periods when the person is sleeping for monitoring SA. There may be differences in the setup as the recording with NOX T3 and the recording with FLOW is not necessarily started simultaneously. Furthermore, both sensors are wireless (e.g., Bluetooth), which means the signal can drop or become weak. A likely scenario is that a person goes to the bathroom in the middle of the night, causing a potential connection loss depending on the signal range.

Accuracy: The accuracy dimension is the degree to which the recorded data correctly describes the real-world entity it represents. Accuracy, in contrast to most DQ dimensions, cannot be evaluated by analyzing the data itself. Instead, measuring the accuracy of recorded data can only be achieved by comparing it to the ground truth, that is, the real-world entity it represents, which in most situations is not practical or even possible. As a result, it is common to use a third-party reference deemed trustworthy enough to be the gold standard. The gold standard should most accurately describe the ground truth and is, in our case, the RIP abdomen signal from the NOX T3.

The accuracy of our dataset in the context of sleep apnea monitoring is ultimately reflected by the classification performance of machine learning classifiers for the detection of disrupted breathing. To do this, we use the annotated NOX T3 scored by a sleep expert from Oslo University Hospital to compare the results of the two signals.

3.2 Signal Quality Metrics

Defining metrics for measuring the signal accuracy of a sensor depends on the nature of the data in question. In the context of sleep apnea monitoring, the most precise method for measuring the signal quality of a respiratory effort sensor is to include it in traditional polysomnography using real sleep apnea sufferers, and let trained personal score the results manually. Alternatively, studies have shown that there is a close agreement between the results from home sleep testing, using the NOX T3 monitor, and the results of in-laboratory polysomnography [Xu et al. 2017]. The disadvantage of this kind of study is the lack of supervision, which means we cannot help patients with proper placement of the sensors. Nevertheless, the recordings are in a real-world setting and reflect what we expect users to record in an initial sleep monitoring test at home.

There are some consequences of relying on the classification performance as the only indication of the signal quality. For instance, if the performance of a classifier is inferior, without any other quality metrics, it is difficult to determine which aspects of the underlying data or recordings are to blame. Some recordings may be of such poor quality that they should not be used to train a model or to evaluate the prediction accuracy. The annotated scoring from NOX T3 must be aligned with the signal from FLOW accurately enough that all SA events correspond to the signal. If this is not the case, the classifier may learn the wrong patterns during training, or even if it predicts correctly, the evaluation may say otherwise if the annotated scoring is not aligned accordingly. As a result, we consider different ways to measure the signal quality.

The accuracy of a physiological time series data is simply the distance between it and the ground truth or gold standard. This distance can be measured in several ways, with one option being a raw signal comparison of the waveforms using a domain function. For this to be meaningful, both signals need to have the same unit of measurement. For instance, if one sensor measures the respiratory process by the changes in airflow and another by the change in lung volume, their signals cannot easily be compared as the measurement units are different. The drawback of a raw signal comparison is that all signal parts are compared even though not everything may be relevant to the context. The accuracy of respiratory effort sensors for sleep apnea monitoring is related to their ability to detect breaths and disrupted breathing. More precisely, the requirement is that such sensors should be able to detect apnea and hypopnea events.

3.2.1 Apnea Detection

The definition of an apnea is given as a complete cessation of breath lasting for at least ten seconds. There are two types of apnea events, central and obstructive, although a combination of them is possible. For instance, an apneic event can start as a central apnea and develop into an obstructive apnea. During an obstructive apnea, the respiratory effort is persistent or increased as the patient is trying to force air through the physical obstruction. In the case of a central apneic event, the patient shows no signs of respiratory effort. Figure 3.1 illustrates a number of different respiratory effort signals during an apnea event. Notice that both the $\text{RIP}_{\text{abdomen}}$ and $\text{RIP}_{\text{thorax}}$ show respiratory effort during the apnea, which means it is an obstructive apnea. As a result, RIP belts are not recommended by AASM for apnea monitoring, but are recommended for determining the type of apnea event as the sufferer will exhibit different respiratory behavior.

According to Løberg [Løberg 2018], the most fundamental quality metric for apnea detection is the breath detection accuracy. Meaning if a sensor can accurately detect breaths, it is trivial to reason about its ability to detect apnea events (i.e., gaps between breaths for more than ten seconds). However, this is not true for respiratory effort sensors like FLOW since the signal may exhibit a different waveform than NOX T3 during obstructive apneas

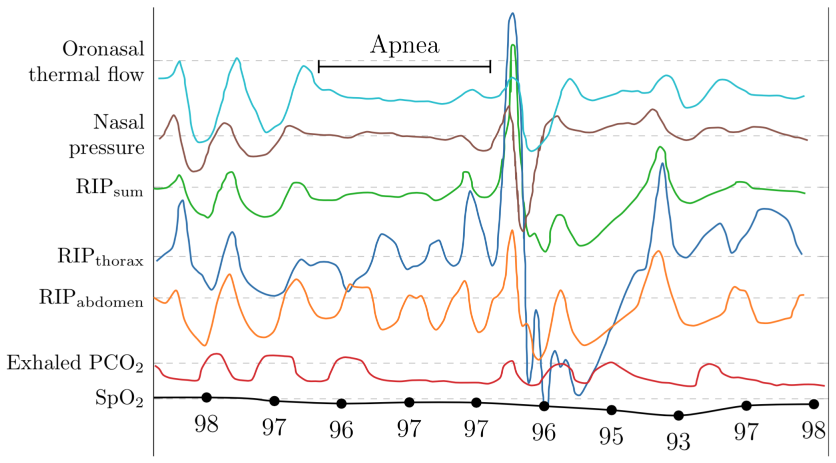

Figure 3.1: Respiratory signals during an obstructive apnea [Berry et al. 2012]

that are not necessarily bad for machine learning classification, but may result in fewer or more breaths detected. Nevertheless, breath detection accuracy should still be useful as an indication of the signal quality.

Breath Detection Accuracy

For measuring the breath detection accuracy of a signal, we use the following metrics: sensitivity, positive predictive value, and clean minute proportion. These metrics suggested by Løberg are described as follows:

Sensitivity: The sensitivity metric reflects the ability to identify breaths correctly. Data from a low cost sensor like FLOW might not reveal all of the breaths that can be detected with the data from NOX T3, or some might be false breaths. One reason for false breaths to occur is if the signal is unable to flatline during central apnea events. The formula for sensitivity is given in Equation 3.1, where $|B_{true}|$ is the number of correctly identified breaths, and $|R|$ is the number of real breaths detected by the gold standard.

$$Sensitivity = \frac{|B_{true}|}{|R|} \times 100\% \tag{3.1}$$

Positive Predictive Value: PPV is used to determine the proportion of correctly identified breaths in relation to all breaths detected. A perfect PPV score of 100% means that none of the detected breaths are false breaths. As such, it yields information about the proportion of false breaths. The formula for PPV is shown in Equation 3.2, where $|B|$ is the total number of detected breaths.

$$Positive\ Predictive\ Value = \frac{|B_{true}|}{|B|} \times 100\% \tag{3.2}$$

Clean Minute Proportion: CMP is used to understand the distribution of errors in the sensitivity and PPV metrics. A minute is considered clean when both the sensitivity and PPV score are 100% during the minute. As such, a perfect CMP score of 100% means that there are no errors in the two metrics for the given number of minutes. See Equation 3.3, where $|M|$ is the length of the recorded signal in minutes, s yields the sensitivity, and ppv yields the positive predictive value. The CMP score is calculated with Equation 3.4, where $|M_{clean}|$ is the number of clean minutes.

$$M_{clean} = \{m \in M \mid s(m) = 100 \; and \; ppv(m) = 100\} \tag{3.3}$$

$$Clean\ Minute\ Proportion = \frac{|M_{true}|}{|M|} \times 100\% \tag{3.4}$$

3.2.2 Hypopnea Detection

For a period of shallow breathing to be considered a hypopnea event, there needs to be a minimum of 30% reduction in airflow lasting a minimum of ten seconds, in addition to either a $\geq$3% drop in oxygen saturation or arousal. Without any other signals than a respiratory effort sensor, we can only consider the reduction in airflow requirement. The breath amplitude (level of airflow) measured by the sensor needs to be linear in relation to the abdominal expansion. As a result, the breath amplitude accuracy is how we want to measure the accuracy of hypopnea detection, with the prerequisite of an accurate detection of breaths.

Breath Amplitude Accuracy

With a linear relationship between breath amplitude and belt distraction, a 30% reduction in breath amplitude corresponds to a 30% reduction in airflow. This relationship can also be monotonic, meaning the sensor measures a 30% reduction in breath amplitude, although the airflow reduction is only 15%. Moreover, the sensor may become somewhat trapped, reducing the detected breath amplitudes even if the airflow remains the same.

We have chosen to measure the breath amplitude similarly to how hypopnea events are scored using an error rate as a percentage. Meaning a breath amplitude error rate of 5% means that the measured reduction in airflow is within 5% of the actual reduction. As previously explained, hypopnea events are scored when the airflow is reduced by $\geq$30% for ten seconds or more. Therefore, a sensor with a 5% amplitude error rate means that all hypopnea events with a minimum of 35% reduction in airflow are identified. The reason is that the sensor will measure a 35% reduction in airflow in the range of 30-40%. Since the two sensors use different scales of measurement, we cannot compare the amplitudes directly. Instead, we obtain the relationship through linear regression as we assume and expect the relationship to be linear. More precisely, we fit a linear regression model to the breath amplitudes of FLOW and NOX. The error rate of one breath amplitude in the FLOW sensor is then calculated as its distance to the regression line [Løberg 2018]. Figure 3.2 illustrates how the regression line is intended to work.

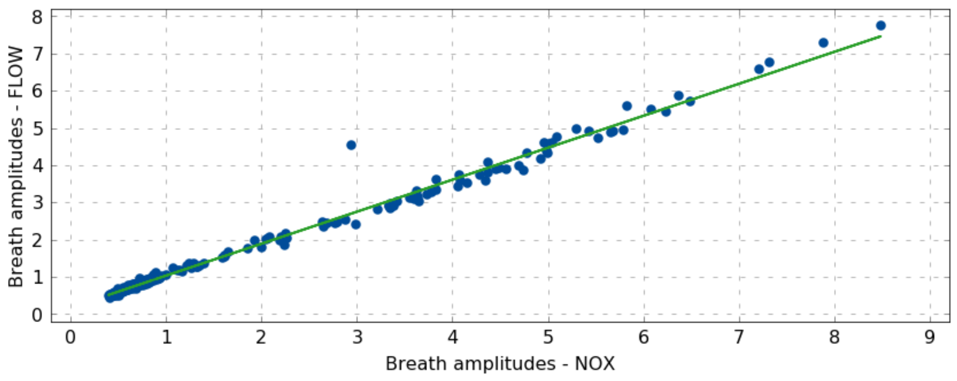

Figure 3.2: Example of regression line

Finally, to summarize the error of all breath amplitudes, we use the *weighted absolute percentage error* (WAPE). The WAPE formula is shown in Equation 3.5, where E are the regression line values and B are the actual breath amplitudes recorded by FLOW, and $\overline{B}$ is the calculated mean value of all breath amplitudes. The WAPE metric works by calculating the error percentages of all breath amplitudes relative to the mean of all breath amplitudes known as the baseline.

$$\begin{aligned} n &= |B| \\ WAPE &= \frac{1}{n}\sum_{i=0}^{n}\frac{|E_i - B_i|}{\overline{B}} \times 100\% \end{aligned} \tag{3.5}$$

3.2.3 Breath Detection

We will utilize the script in [Løberg 2018] to automatically detect breaths and calculate the sensitivity, PPV, CMP, and WAPE metrics. The breath detection is based on a form of peak detection constrained by physical limitations. The limitations are needed as not all peaks are breaths. Therefore, the following requirements are used to determine if a peak should be regarded as a breath:

- The duration must last between 0.6 and 12 seconds.
- The amplitude of the peak must be at least 10% of the mean breath amplitude.
- Breaths cannot overlap.

This script can be found in Appendix A, with some modifications to suit our needs, as we explain later in Section 5.3.

3.3 FLOW Issues

Løberg [Løberg 2018] evaluates the signal quality of FLOW amongst other sensors for sleep apnea monitoring in a controlled and surveyed environment. During the signal capture sessions, the subjects performed several tasks to simulate sleep apnea events, for example, holding their breath or breathing shallow for an extended period of time. Since the subjects were awake the whole time, only shorter sessions of about twenty minutes were conducted. The controlled environment is the main reason that they did not experience any connection issues with the FLOW sensor. Moreover, the unreliable timestamps were easily adjusted due to the short duration of the capture sessions.

To understand why we experience these issues in a real scenario, we look at what makes our set-up scenario different. One aspect is that we have longer recordings lasting typically between seven or eight hours. We conduct our sessions in the home of patients without any supervision where patients are sleeping with both the NOX T3 sleep monitor and the FLOW sensor for the entire night. We have no control over the sleeping positions of the patients, which likely change several times throughout the night. Besides, they may wake up to go to the bathroom in the middle of the night, in which case we have no information about it. If the patient with the attached sensor is too far away from the smartphone, the Bluetooth connection may be weakened or lost. The signal may also become blocked if a patient sleeps on their stomach. Moreover, changing sleeping position may cause the belt to become misaligned or trapped altogether. We have not one but several FLOW sensors at our disposal, which means that we can get a better insight into the consistency of quality for the different FLOW sensors.

3.3.1 Connection Loss

One of the first issues we discovered with FLOW is that the sensor tends to loose connection sporadically while recording. An example of this can be seen in Figure 3.3, where the signal drops for around ten seconds before it resumes at another baseline. This issue is very common since most of the overnight recordings in the dataset have one or more connection losses.

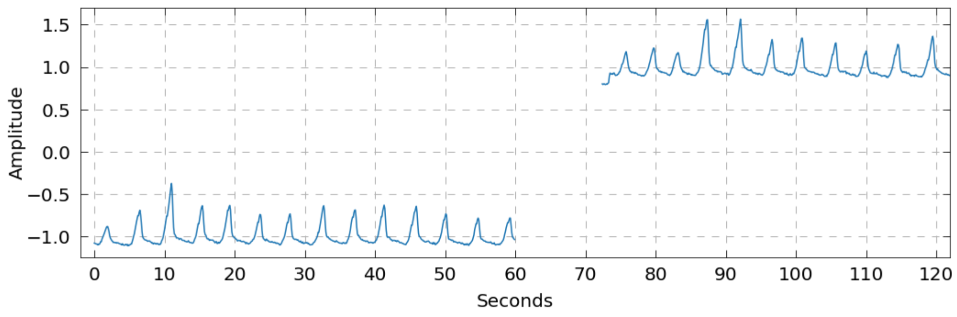

Figure 3.3: Example of signal drop during recording

After inspecting several recordings, the first impression is that gaps around five seconds or more between two subsequent samples are the result of a loss of connection. For reasons that will become clear in Section 4.1, we cannot generalize this or be completely sure. Instead, we try to estimate the number of connection losses and their duration in each recording based on gaps longer than five seconds.

The results of this estimate are shown in Table 3.1, where the recordings are grouped based on the specific FLOW sensor used to record. The eight FLOW sensors we use are listed as A, B, C, E, F, G, H, and I. The table lists the number of recordings, the total duration, how many gaps that are longer than 5 or 60 seconds, and the total signal loss. One may first notice that all sensors have periods of connection loss. Sensor C has lost the most data according to this estimate, but the total recording time for this sensor is significantly higher than many other sensors. The signal loss percentage may be better for describing the connection issues of a sensor. For instance, sensor H has a signal loss rate of 17.5%, indicating that it has more connection issues than sensor F, which only has a loss rate of 1.0%. However, this is not so obvious when we consider that one of the recordings by sensor I has lost more than six hours of data, while the remaining five recordings have lost eleven minutes combined. Nevertheless, there is a correlation between the signal loss rate and the total number of gaps longer than five seconds. Sensors with a high loss rate such as sensor C, sensor I, and sensor H have more gaps, compared to the remaining sensors.

Each patient is expected to sleep for around six to eight hours, depending on their sleeping habits, but we discovered that several recordings stop in the middle of the night for unknown reasons. One possible explanation is that the application on the smartphone crashed, which would make sense since this problem is not limited to a specific sensor. This assumption is based upon our experience with the application. This application creates a summary file when the user presses stop recording. None of the recordings that end in the middle of the night have such a summary file, which indicates that something went wrong.

To summarize, all sensors have connection issues. Although it may seem that this problem is more apparent for individual sensors, the limited sample size of each sensor means that a single recording with severe connection issues can significantly impact the overall results of a sensor. To understand the exact cause behind the connection losses, one would have to study the individual FLOW sensors in a controlled environment such as a laboratory.

3.3.2 Unstable Sampling Rate

The benefit of using eight individual FLOW sensors during the data acquisition phase is that it allows for a broader and more general analysis of the quality of the sensor. SweetZpot states that the FLOW sensor samples at 50

Sensor	Rec.	Duration	Gaps > 5s	Sig. loss	Loss (%)
A	11	71h	37	58m	1.3%
B	7	39h	61	33m	1.4%
C	9	62h	119	460m	11.0%
E	10	75h	74	165m	3.5%
F	5	23h	14	14m	1.0%
G	4	25h	65	103m	6.4%
H	8	33h	108	419m	17.5%
I	6	36h	175	385m	15.1%

Table 3.1: Recording distribution based on sensors

Hz, but averages five subsequent samples as they arrive internally, before it outputs a signal at 10 Hz. However, our analysis shows that for most recordings, the sampling rate is not close to being precisely 10.0 Hz.

The boxplots with the distribution of sampling rate for every recording can be seen in Figure 3.4 grouped by sensor ID. We have chosen to include two versions of the sampling rate because of the connection issues. Figure 3.4a shows the sampling rate calculated using the number of samples and the duration using Equation 3.6. In Figure 3.4b, we estimate the sampling rate minute-wise using Equation 3.7, where m is the total number of minutes in the recording and S_i is the number of samples in a specific minute. We exclude any minutes without samples and average the sampling rate of all minutes, resulting in the estimated sampling rate.

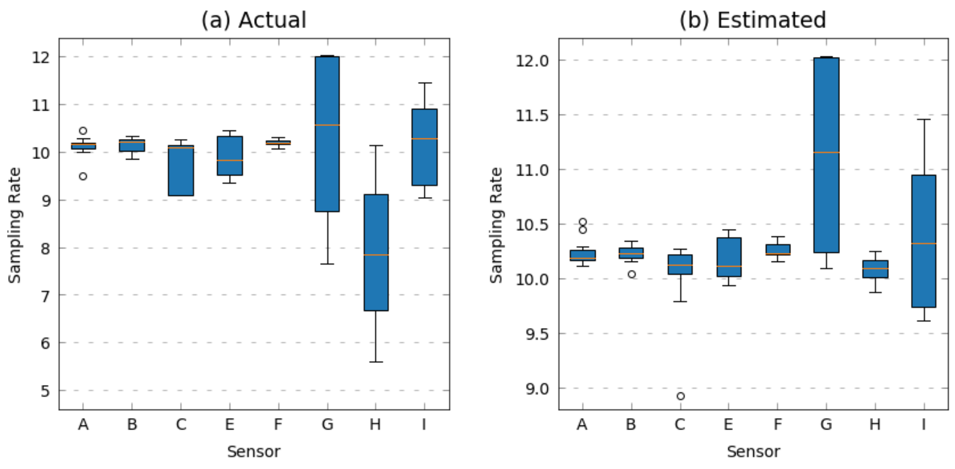

Figure 3.4: Actual and estimated sampling rate for sensors

$$Hz = \frac{samples}{duration\ (sec)} \tag{3.6}$$

$$Hz_{est} = \frac{1}{m}\sum_{i=0}^{m}\frac{S_i}{60}\ ,\ for\ S_i\ >\ 0 \tag{3.7}$$

As seen in Figure 3.4a, there is a significant difference in the sampling rate between some sensors and even among recordings with the same sensor. We expect that some recordings have a lower sampling rate than 10 Hz, given the connection issues, but surprisingly some recordings have a sampling rate of around 11-12 Hz, as seen in the boxplots for sensor G and I. Moreover, when we consider the connection issues, the estimated sampling rate in Figure 3.4b, indicates that the sampling rate for most recordings is more than 10 Hz. Generally, the sampling rates seem to be in the range of 10.0-10.5 Hz. The only exception are sensors G and I, which have a higher variation. The FLOW presumably samples at 10 Hz, but our findings show that all sensors over-sample, that is, all sensors have recordings with a sampling rate higher than the stated 10 Hz. One possible explanation is that FLOW does not sample precisely at 50 Hz internally, assuming that the process of averaging the samples are done correctly. This is not a problem in itself, but the fluctuating sampling rate raises a concern about the reliability and quality of the hardware and product.

It is concerning that the sampling rate is not always the same across sensors and that it can change between recordings with the same sensor. Furthermore, it raises the concern that the sampling rate can change during recording. We have chosen to inspect the sampling rate for a few selected recordings from four sensors, as seen in Figure 3.5. This selection is based on some of the variations we have observed in the sampling rate. The sampling rate is estimated minute-wise and excludes any minutes without data. Finally, the sampling rate of ten subsequent minutes are averaged to reduce clutter. Each subplot shows two recordings from that specific sensor. One may first notice how similar and stable the sampling rates are for the recordings from sensors C and E in Figure 3.5a and Figure 3.5b, respectively. The few sudden drops below 10 Hz are most likely the result of one or more small connection losses. For example, given a sample rate of 10 Hz across nine minutes, if the sampling rate of the last minute is estimated to 1 Hz, the average sampling rate for the ten minutes is calculated as: $\frac{9 \times 10Hz + 1Hz}{10min} = 9.1Hz$.

The recordings from sensor G and I have more variation, as seen in Figure 3.5c and 3.5d. Both have recordings with a sampling rate more or less stable just above 10 Hz, but also recordings with an unusual high and unstable sampling rate. It is interesting that the recordings with a higher sampling rate change more during recording, than those with a sampling rate around 10 Hz. After a visual inspection of the signals, there is no indication that the variations are a result of connection issues. Our analysis indicates that the sampling rate of FLOW is not necessarily stable during recording.

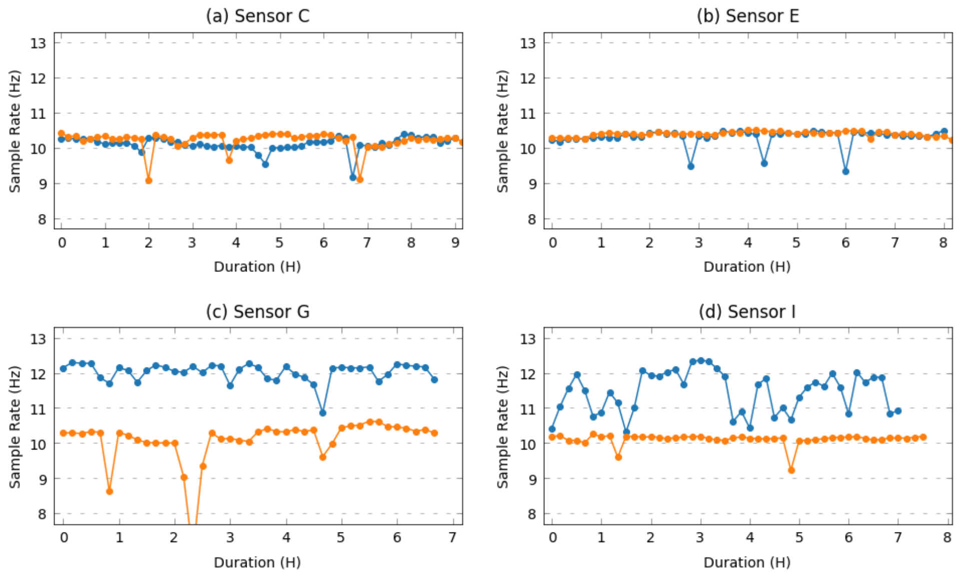

Figure 3.5: Sensor sampling rate variation

3.3.3 Unreliable Time-stamping

So far, we have strong indications that the FLOW sensors do not output a signal at 10 Hz precisely. This may not even be a problem, if not for the fact that the timestamping of samples in a recording also is unreliable. To illustrate what we mean by this, we plot in Figure 3.6a the samples on the y-axis represented as blue dots and their corresponding timestamps on the x-axis. Additionally, we plot in Figure 3.6b the same samples with the only difference being that we have changed the timestamps by separating each sample with 100 milliseconds. The difference is very obvious, as the samples in Figure 3.6a are clustered together in groups with almost identical timestamps, while the line drawn between subsequent samples in Figure 3.6b looks like a sinus-curve, typical for how respiratory effort sensors measure breathing.

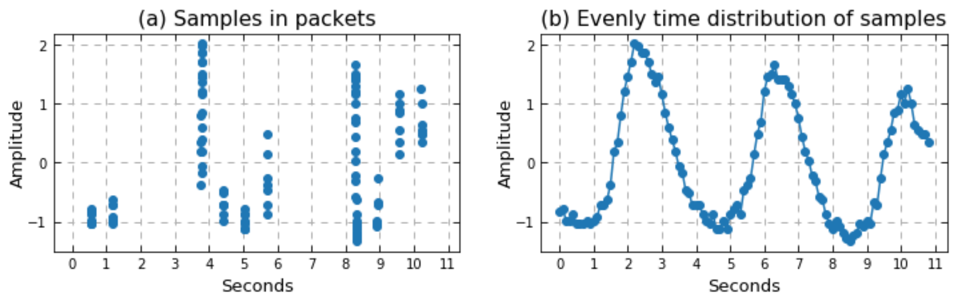

Figure 3.6: Example of timestamping FLOW samples

Presumably, what happens is that when the Sleep Recorder application receives a packet (via Bluetooth), it timestamps each sample as they are read.

This assumption is based upon the fact that the inter-sample arrival time (according to their timestamps in the output file of the application) has a high variance where the expected 100 milliseconds is a value that seldom or never appears. In most cases, however, two consecutive samples have almost the same timestamp. Besides this, 700 ms between two timestamps appear rather often, which we attribute to the possibility that seven samples are collected in one Bluetooth packet. There are sometimes even longer inter-sample arrival times, which we attribute to the possibility that different Bluetooth packets are not always handled immediately. For instance, the smartphone could be busy with doing something else, which means the packet ends up in a queue somewhere in the kernel/OS until the process running the Sleep Recorder application starts again.

The sleep recorder application on the smartphone often handles more samples at a time when it has not processed any for a few seconds, as seen around the four-second mark in Figure 3.6a. Since several packets may be queued before being processed quickly after another, it is difficult to know the exact size of every packet based on the timestamping. Instead, what we can determine with some certainty, is how many samples the application handles/processes at a time. This assumption is based upon the fact that many samples have identical timestamping, which means that if there are even a few milliseconds between two subsequent samples, we believe that the latter sample is part of a new packet. We use a margin of ten milliseconds between two subsequent samples as the threshold for deciding when the following sample is not part of the current round of processing.

The results are presented as a pie-chart in Figure 3.7. For half of the time, the application processes seven samples, which further indicates that this is the typical size of a packet. However, for the remaining 50% of the time, the number of samples the application processes is more disperse, indicating that the packet size is not fixed, but for some unknown reason, it usually consists of seven samples.

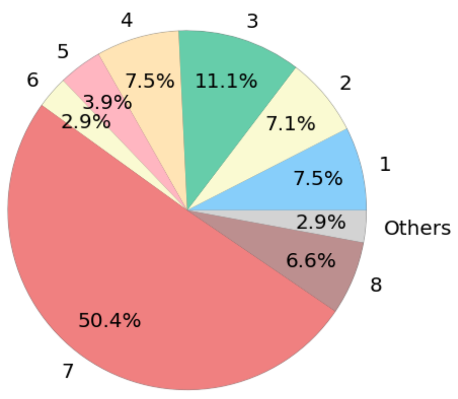

Figure 3.7: Number of samples processed in a batch by the Sleep Recorder app

It is also interesting to consider how often the sensor sends a packet via Bluetooth or how often the application runs the processing. As such, we group the samples by timestamps in ranges of 100 milliseconds to get an approximate idea of the distribution. We present the results in a pie-chart in Figure 3.8. 71% of the time, there are 600 -699 milliseconds between two rounds of processing, which highly correlates to the fact that seven samples are combined into one packet, causing this down-time between each round where the application has nothing to process. Since seven samples are often processed together, it is interesting that there is seldom 700-799 ms between subsequent processing rounds. This is another indication that the sampling rate exceeds 10 Hz, or else one would expect seven samples to be processed every 700 milliseconds more or less.

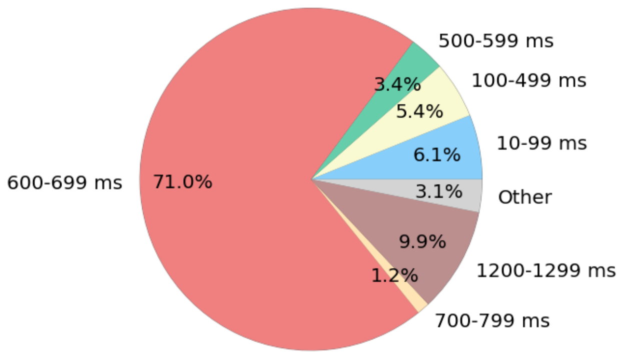

Figure 3.8: Time between processing runs in the Sleep Recorder app

3.3.4 Baseline Issues

The definition of baseline in the context of respiratory effort signal is the value on the y-axis between breaths, or more precisely, where a breath starts and ends. For longer recordings, the baseline is not necessarily stable for several reasons. First of all, the baseline of respiratory effort signals tends to wander, meaning the value on the y-axis between breaths slowly increases or decreases as time passes. More abrupt changes in the baseline are often related to movement by the subject. The reason is that respiratory effort sensors capture any sort of motion and not only motion related to breathing. The signal noise related to movement is commonly referred to as motion artifacts. As shown in Figure 3.9, the signal from FLOW struggles with both motion artifacts, baseline wander, and baseline shifts. In this particular case, the baseline wander is highest during the first hour of recording, but it is present for the majority of the signal. In some parts, the direction of the baseline wander changes after a baseline shift. The baseline shifts are often preceded by large motion artifacts, possibly because the patient changes their sleeping position.

Reducing or removing baseline wander in signals such as ECG, is often

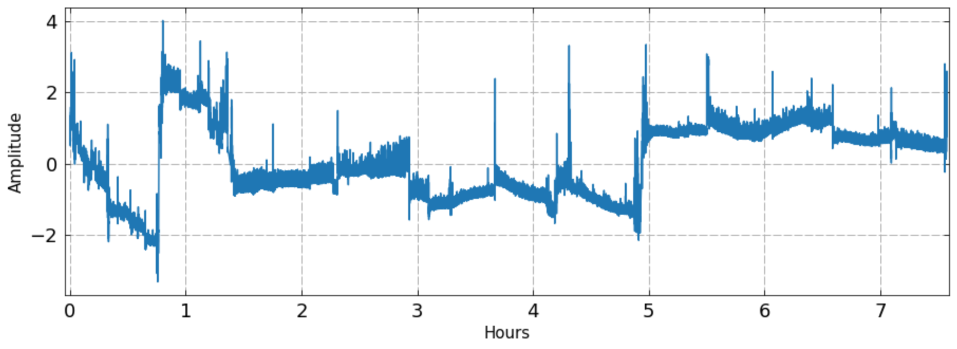

Figure 3.9: Overnight FLOW recording

a requirement. Depending on the domain and severity of the baseline wander, different approaches can be useful. Since baseline wander is mostly low-frequency noise, one option is to use a high pass filter, but at the risk of attenuating the respiratory component, if not done correctly. Another option is to fit a low-order polynomial to the signal, which essentially is finding the trend or the signal behavior over time and subtracting it from the signal. This approach is a simple technique that leaves the respiratory component mostly intact.

In the case of baseline shifts, the change is more abrupt to such a degree that fitting a low-order polynomial is not sufficient. As this is related to movement, one option is to use accelerometer data to identify motion artifacts. Virtanen et al. [Virtanen et al. 2011] implement an accelerometer-based motion artifact removal (ABAMAR) algorithm that can be used for such cases. If the acceleration (movement) between two consecutive samples exceeds a threshold, the algorithm flags the corresponding time interval T_m as a motion artifact. In cases of prolonged movement, T_m can be extended accordingly. Not all motion artifacts change the baseline. Therefore, the mean amplitudes of the signal before (A_{before}) and after (A_{after}) the motion artifact are compared. When the mean amplitude difference exceeds a threshold based on the standard deviation (SD) of A_{before}, a baseline shift is detected. To correct for the baseline change, the baseline of the preceding period is imposed on the signal after, by multiplying the amplitudes after T_m by $\frac{A_{before}}{A_{after}}$. A margin of some seconds around the motion artifact is used to avoid noise interfering. As such, to remove the transient motion artifact, the signal T_m and the margin are set to A_{before}.

We illustrate in Figure 3.10 how this algorithm can be customized to work on the signal from FLOW. The accelerometer data from NOX T3 has flagged the period T_m as a motion artifact. The margin around T_m ensures that the baseline is not affected by the motion artifact. A smaller margin works well for A_{before}, but A_{after} needs a larger interval as it takes some time before the signal stabilizes at a new baseline. In other words, the accelerometer detects the motion artifact correctly, but T_m does not cover the entire period where the signal is noisy. How long it takes the sensor to calibrate to the new baseline after a shift may be different from time to time. It might be challenging to chose a T_m that is large enough without removing unnecessary amounts of data.

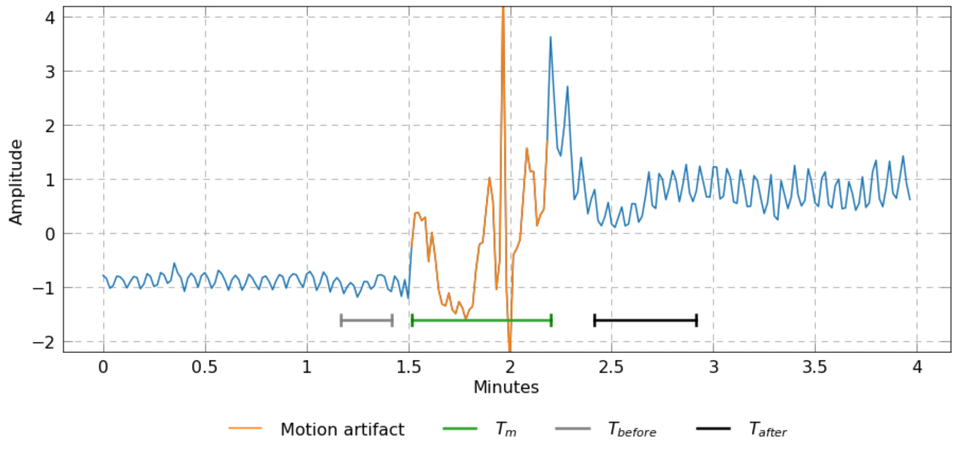

Figure 3.10: Example of ABAMAR algorithm detecting baseline shift

3.4 Discussion and Conclusions

There are four major types of issues regarding the initial quality of FLOW recordings, summarized as follows:

- **Connection loss**
  The signal from FLOW tends to drop for shorter or extended periods of time. Deciding if an arbitrary gap between the timestamps of two consecutive samples is the result of a connection loss is dependent on the reliability of timestamps and possibly the sampling rate.

- **Unreliable timestamping**
  The timestamping of samples, indicates that samples are timestamped by the Sleep Recording application as soon as they arrive. Although seven samples often have almost identical timestamps, longer inter-sample arrival time between consecutive samples indicates that packets are sometimes queued somewhere in the smartphone.

- **Sampling rate variation**
  The sampling rate of FLOW is not precisely 10 Hz, as stated by SweetZpot. The sampling can change from one recording to another and during recording. The sensor tends to slightly over-sample, but the combination of connection loss and unreliable timestamping makes it difficult to measure the sampling rate precisely.

- **Baseline wander and shifts**
  For overnight recordings, FLOW has a large number of baseline shifts,

which are likely related to movement by patients as they turn and changes sleeping position.

Chapter 4

Preprocessing

In this chapter, we describe the challenges we face during the preprocessing of FLOW recordings. Based on the data quality issues discussed in the previous chapter, we have identified four particular tasks needed for preprocessing sensor data from FLOW:

- connection loss detection and ascertaining when a gap is not caused by jitter,
- time stamp adjustment,
- start time of recording and start time after a gap, and
- sampling rate measurement.

Additionally, tasks to extract and export the NOX T3 abdomen signal are needed, before both signals can be combined and synchronized to measure the quality of FLOW recordings.

We begin Section 4.1 with a brief description of the data issues experienced with the FLOW sensor and how we can detect connection loss. We continue in Section 4.2, with an overview of several design alternatives for adjusting the timestamps of FLOW, along with a discussion of their suitability. In Section 4.3, we describe the process of estimating timestamps, followed by how the sampling rate can be measured in Section 4.4. Next, we present the results of several preliminary tests in Section 4.5. In Section 4.6, we describe the system environment used during preprocessing, before we define several automatic scripts for preprocessing the dataset in Section 4.7. This includes the process of converting and extracting NOX signal data from the European data format (EDF) and the changes made to the existing preprocessing script we use to combine and synchronize the signals. Finally, we summarize and conclude this chapter in Section 4.8.

4.1 Connection Loss Detection

Many of the FLOW recordings contain periods without any respiratory data, and the application used for capturing the sensor data has a rather high variance in inter-sample packet arrival time. Due to these issues, we want to distinguish between connection losses and gaps in the recording caused by jitter. However, it turns out that the distinction is not trivial. To explain this challenge, we illustrate in Figure 4.1 and Figure 4.2, the differences and similarities between connection loss and jitter. In these figures, the red line represents time, the blue line represents Bluetooth connection between the sensor and smartphone, and the blue squares represent when a packet, with several samples from the sensor, arrives at the application on the smartphone.

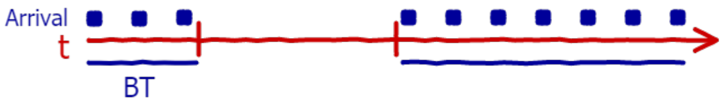

Figure 4.1: Example of connection loss

In Figure 4.1, the first three packets arrive at a constant rate before a long inter-arrival time before the fourth packet arrive. The following packets arrive again at a constant rate as the ones before the gap. At first, it seems obvious that there is a connection loss as indicated by the blue line and as a consequence, a loss of one or more packets, but as Figure 4.2 shows, this is not necessarily the case. After the first three packets arriving at a constant rate there is a *gap*, i.e., for a longer time period no packets arrive. At the end of this period multiple packets arrive almost at the same time. Similar to the situation in Figure 4.1, the following packets arrive again at a constant rate as the ones before the gap. In this case, we have no connection loss as indicated by the continuity of the blue line. Instead, four packets arrive at a later time than expected. If we place these packets evenly spaced in the longer time period without packets, there are no indications of packet loss. This type of jitter can have several causes, such as buffering on the sensor or somewhere on the smartphone. The signal can also become weak or temporarily blocked due to an increase in distance between the sensor and smartphone.

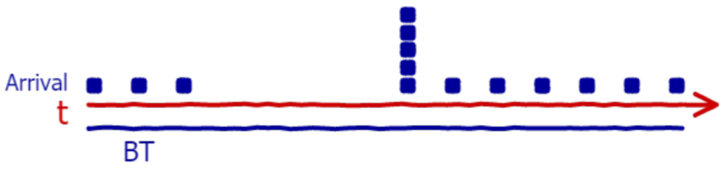

Figure 4.2: Example of jitter

If jitter causes the packets to bundle up, we can separate gaps caused by jitter from gaps caused by connection loss. To do this, we check if the packets that arrive directly after the gap contain the packets we were expecting to arrive during the time period of the gap (assuming a constant packet arrival rate). Unfortunately, it is more complicated than this. As Figure 4.3 shows, sometimes the missing packets arrive at a much later time. Besides, there is not a single large gap, but rather a high variance in the inter-arrival time of packets. In such cases, it is difficult to know if there is a loss of connection or for how long we need to wait for the missing packets to arrive.

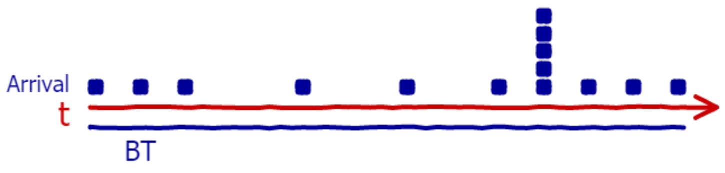

Figure 4.3: Example of high variance in inter-arrival time of packets

To fully understand the scale of the problem we are facing with jitter and why it is difficult to detect connection losses, we have used a short segment of a real recording, as seen in Figure 4.4, to show how late packets sometimes arrive compared to the estimated time they were recorded. In this figure, we have the recording time in seconds on the x-axis, while the y-axis represents the strain of the sensor, which changes as the patient breathes. The strain values are standardized and denoted as amplitude since it represents the amplitude of breaths. We have also included the time that packets arrive and how many packets are processed at a time in Figure 4.4. Each packet contains seven samples in general, although some variations exist.

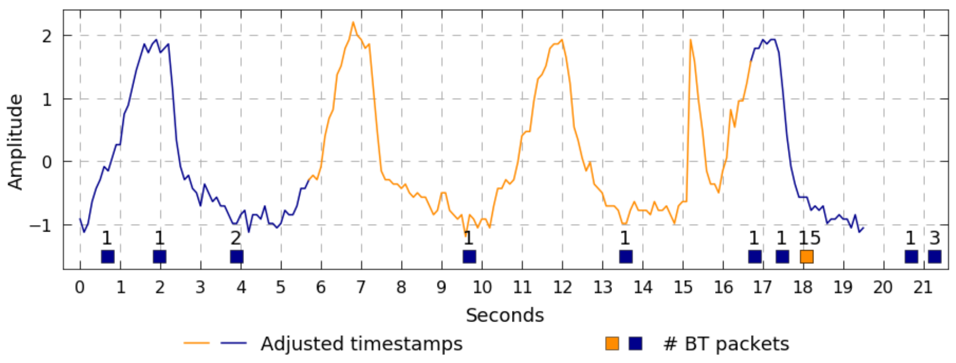

Figure 4.4: Example of jitter in a real recording

In this example, packets have a high variation in inter-arrival time, where the most notable one is the large arrival of 15 packets highlighted in orange. Due to this jitter, the application adds incorrect timestamps to the samples. However, adjusting the timestamps, using the expected interval of one sample every 100 milliseconds, reveals a typically looking respiratory signal without gaps. Interestingly, the samples in the orange packets (highlighted by the orange line) get distributed over a 10 seconds time span. The late arrival time of packets compared to the estimated time they were recorded by the sensor, makes it significantly harder to detect connection loss without falsely detecting cases, such as this one, where the missing packets arrive later than expected. While it may seem obvious that larger gaps of minutes or even hours are caused by connection losses, we cannot be completely certain without visually inspecting the FLOW signal and comparing it with NOX.

4.2 Time Stamp Adjustment

As described in Section 3.3.3, our early analysis of the FLOW sensor has shown that multiple samples are collected in one Bluetooth packet and then send to the smartphone. The arrival time of packets in the app determines when the app is able to timestamp the tuples in the packet(s) with the current time. The ideal solution is that the sensor timestamps each sample considering we have no control over how the operating system on a smartphone handles the packets. However, SweetZpot has not provided us with a reason why this is not the case. Most likely, it is not necessary for the contexts they usually work with, such as rowing or cycling.

To address this issue, we have to develop a way to adjust the timestamps of FLOW. The specificity and characteristics of our problem means that it is not easy or necessarily possible to find the optimal solution through a study of literature. Instead, we use a more hands-on approach of trying five different alternatives that seem to fit our case the best. The selection is based on our analysis and the knowledge from working with the data from FLOW. We analyze the timestamp adjustment of different alternatives by synchronizing the signal with NOX for comparison. The idea is to start with the most simple approach. If this approach falls short of solving our problem, we should have more information helpful for finding other solutions.

The first alternative is to rely on the information from SweetZpot that claims the sensor produces samples with a constant rate of 10 Hz. A sampling rate of 10 Hz means there should be exactly one sample every 100 milliseconds. Then given the start time of a recording, all timestamps can be updated such that the first tuple of the recording gets as timestamp the start time of the recording, and the timestamps for all subsequent tuples get as timestamp, the previous tuples timestamp, increased with 100 milliseconds. An illustration of how the timestamps are adjusted with this approach is shown in Table 4.1, where the first packet is processed 602 ms after the start time of recording by the application. The first sample gets the start time of 0 ms as timestamp, and the timestamps of the remaining samples are 100 milliseconds apart.

The second alternative we present is a consequence of the first one. If we cannot rely on the stated sampling rate by SweetZpot, we can try to estimate at what rate the sensor produces samples. The timestamps can then be updated with the same approach as in the first alternative, except using the estimated sampling rate instead. An example of this is shown in Table 4.2 to illustrate the difference between this and the first alternative. In this example, the sampling rate is estimated at 12 Hz. The timestamp of the first sample is again set to the start time of recording, and the timestamps of

Sample	Original timestamps	Adjusted timestamps
1	602 ms	0 ms
2	602 ms	100 ms
3	603 ms	200 ms
4	604 ms	300 ms
5	603 ms	400 ms
6	605 ms	500 ms
7	605 ms	600 ms

Table 4.1: Example of how to adjust timestamps with 100 ms interval

the remaining samples are 83.33 milliseconds apart.

Sample	Original timestamps	Adjusted timestamps
1	602 ms	0 ms
2	602 ms	83 ms
3	603 ms	166 ms
4	604 ms	249 ms
5	603 ms	333 ms
6	605 ms	416 ms
7	605 ms	499 ms

Table 4.2: Example of how to adjust timestamps with 83.33 ms interval (12 Hz)

For the third alternative, we estimate the sampling rate in intervals. This is based on our findings in Section 3.3.2, that the sampling rate is not necessarily constant while recording. The timestamps can then be adjusted with a similar approach as before, using the specific sampling rate for each interval instead. Table 4.3 shows how the samples in the first hour of recording are adjusted using 10 Hz, while the samples in the following hour are adjusted using 12 Hz.

Sample	Original timestamps	Adjusted timestamps
1	602 ms	0 ms
2	602 ms	100 ms
3	603 ms	200 ms
...	...	...
36000	3600608 ms	3600000 ms
36001	3600609 ms	3600083 ms
36002	3600609 ms	3600166 ms

Table 4.3: Example of how to adjust timestamps using intervals with different sampling rate

The fourth alternative we present uses a different approach. If we assume that the timestamp of the last sample in a packet is more or less correct, the timestamps of the remaining samples in a packet can be adjusted using this information. The last sample in a packet should be the least affected by sensor buffering as it is measured right before the packet is sent to the application. Table 4.4 illustrates how the timestamp of sample number 7 of 609 ms decides the timestamps of the remaining samples in the packet. In this example, the samples are adjusted *backwards in time* in intervals of 100 ms. The last alternative is to adjust timestamps in intervals using a window

Sample	**Original timestamps**	**Adjusted timestamps**
1	602 ms	9 ms
2	602 ms	109 ms
3	603 ms	209 ms
4	606 ms	309 ms
5	607 ms	409 ms
6	607 ms	509 ms
7	609 ms	609 ms

Table 4.4: Example of how to adjust timestamps using the timestamp of the last sample and 10 Hz

algorithm inspired by the landmark and sliding window models typically used in real-time data stream processing.

To summarize we have the five following design alternatives:

1. Assume the sensor produces samples with a constant rate of 10 Hz based on the given information from SweetZpot.
2. Estimate the sampling rate for each recording.
3. Estimate the sampling rate periodically throughout each recording.
4. Assume the timestamp of the last sample in each packet is correct.
5. Use a jumping window, which is a variation of the landmark and sliding window approach.

4.2.1 Adjust Timestamps Using 10 Hz Sampling Rate

The first alternative to timestamp adjustment is the solution used in [Løberg 2018]. As described earlier, this study only includes shorter FLOW recordings without sampling rate abnormalities or connection loss. In which case, the following solution proves sufficient for adjusting the timestamps. Given the sampling rate and start time of a recording, all timestamps can be updated using Equation 4.1, where ts_0 is the timestamp of the first sample (i.e., the start time of the recording), n is the sample number, and *interval* is the sampling rate converted to the interval. To convert the sampling rate from Hz to an interval we use Equation 4.2.

$$ts_{adjusted} = ts_0 + sample_n \times interval \tag{4.1}$$

$$interval = \frac{1000}{sampling\,rate\,(Hz)} \tag{4.2}$$

When using this approach on our dataset, the precision of synchronization between FLOW and NOX is accurate at the beginning of the recording, as seen in the first part of Figure 4.5 where the breathing patterns in the signals occurs at the same time. However, in this example, it takes less than a minute before the signals are no longer synchronized. The FLOW signal slowly starts to drift as the sampling rate used to adjust the timestamp is not the actual sampling rate of the sensor. In this particular case, the breaths in the FLOW signal are *extended* compared to the breaths in the NOX signal, which means the actual sampling rate is higher than the 10 Hz we use to adjust the timestamps. Even though a minuscule difference between the sampling rate used to adjust the timestamps and the actual sampling rate will not have an impact on the early phase of a recording, it is enough to affect the precision of synchronization between FLOW and NOX later on. If the sampling rate varies just 0.1 Hz, it is enough to reduce or extend the recording with one second every hour. For each hour of recording, the difference increases, which can affect the synchronization precision towards the end of the recording.

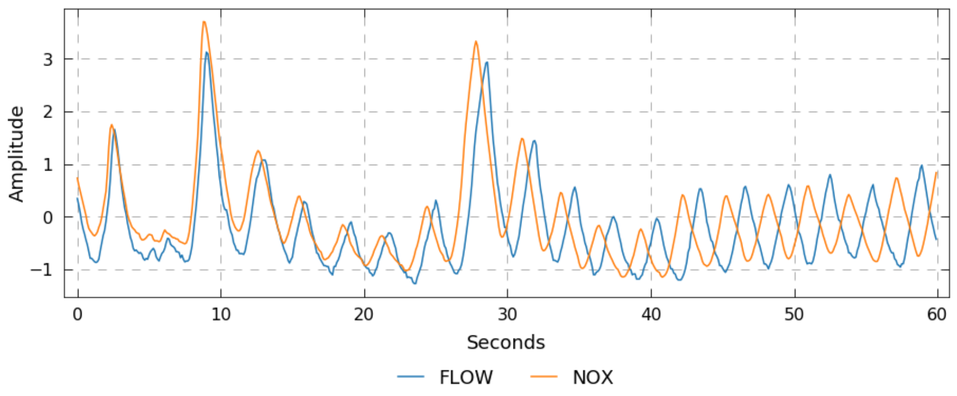

Figure 4.5: Example of adjusted timestamps using 10 Hz

4.2.2 Estimate Sampling Rate

Based on the results from the previous solution, for the next possible solution, we try to estimate at what rate the FLOW sensor produces samples. Using the estimated sampling rate of 10.3 Hz for this recording, we are only able to keep the signals synchronized slightly longer, as seen in Figure 4.6. The figure shows how the FLOW signal starts to drift towards the end of this specific period of the recording. This time, the FLOW signal drifts in the other direction compared to the FLOW signal in Figure 4.5. This indicates that the estimated sampling rate used is too high, or that the sampling rate is not constant throughout the recording. Nevertheless, this alternative is a better solution than the first alternative as it keeps the signals synchronized for a longer time. Moreover, it may work well if we can estimate the sampling rate precisely enough, assuming it is constant.

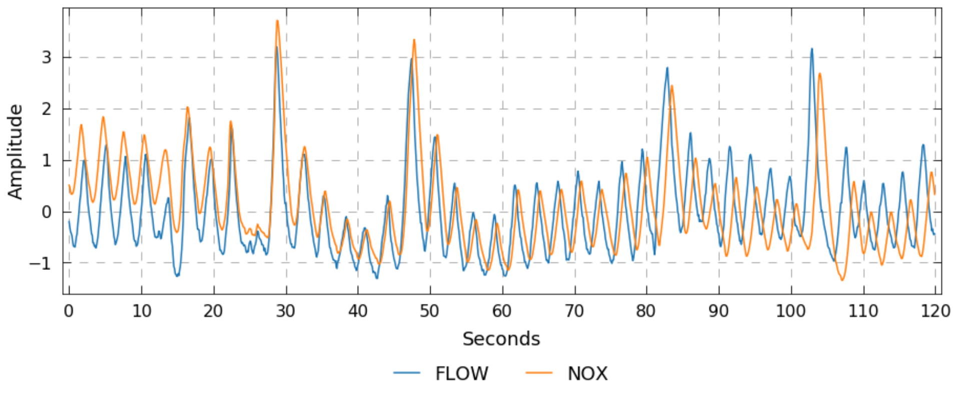

Figure 4.6: Example of adjusted timestamps using estimated sampling rate of 10.3 Hz

4.2.3 Estimate Sampling Rate Periodically

The next alternative we try is to estimate the sampling rate periodically throughout the recording. The estimated sampling rate for each period (or interval) is used to adjust the timestamps with the same approach as the first two alternatives. Proceeding with the recording interval from Figure 4.6, the estimated sampling rate for this two-minute interval of recording is 10.25 Hz. As seen in Figure 4.7, adjusting the timestamps using the sampling rate of 10.25 Hz, keeps the signals synchronized for the duration of the two-minute interval.

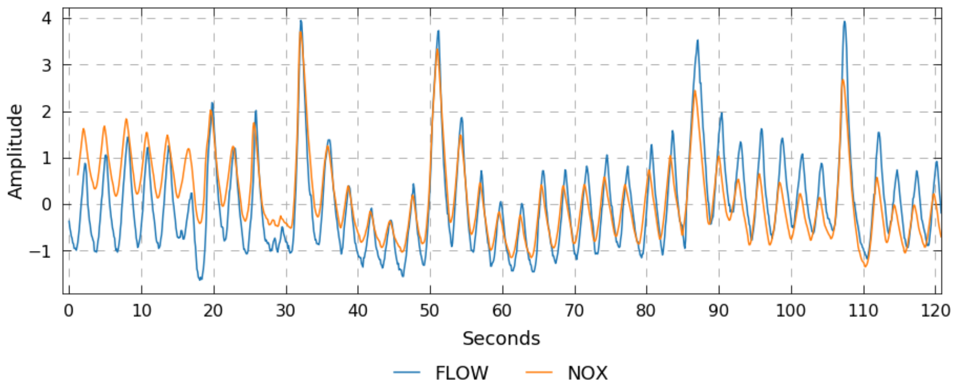

Figure 4.7: Example of adjusted timestamps in interval using estimated sampling rate of 10.25 Hz

Based on the result, this alternative appears to be a promising solution for dealing with an inconsistent sampling rate. However, some issues still remain. Depending on how often the sampling rate changes and to what extend, affect how often we need to estimate it. In the previous example, a two-minute interval is small enough, but if the sampling rate changes drastically all of a sudden, even a two-minute interval may not be small enough. More importantly, this solution cannot solve the connection loss issue. The reason is that we cannot know for sure, as explained in Section 4.1, if an arbitrary period of time without samples is caused by a connection loss or jitter, as the following period may contain the missing samples. Furthermore, it is difficult to estimate the sampling rate accurately for recordings with connection losses.

4.2.4 Adjust Timestamps based on Packet Arrival Time

Instead of using the sampling rate to adjust the timestamps, another option is to adjust them using the already existing timestamps. Since the arrival time of packets determines when the app is able to timestamp the tuples in the packet(s) with the current time, sequential samples in a packet often get identical timestamps as the processing time of samples is minimal. The idea is that the timestamp of the last sample in a packet should be similar to the actual time it was measured by the sensor, assuming the arrival time of a packet is not delayed by jitter, which means the application receives the packet as soon as it reaches the smartphone. Compared to the remaining samples in the packet, the last sample is not queued for as long, since it is the last sample to be measured and collected in the packet before it is sent to the smartphone. The challenge with this alternative is that we cannot trust the timestamp of the last sample in every packet, as jitter can cause the packet to arrive later than expected.

4.2.5 Window Timestamp Adjustment

The last alternative we present is a variation of the landmark and sliding window models, sometimes referred to as the *jumping window* or *moving landmark window*. The general idea is to use an adaptable approach that adjusts the timestamps only when we have enough samples in the window. The three different types of window algorithms are illustrated in Figure 4.8. The start-point of the jumping window is fixed, similar to the landmark window, while the end-point increases, similar to the sliding window, as we look for more samples. The assumption is that if we do not have enough samples in a given window, then extending it may include new data tuples in the window. If we have enough samples in the current window, we adjust the timestamps by evenly distributing the samples between the start and end-point. However, when we do not have enough samples, we cannot keep extending the window forever. Therefore, we set a maximum size that limits how far the window can be extended. A loss of connection is detected if the window reaches the maximum size and still does not contain enough samples. In this case, we do not extend the window further. Instead, the window *jumps*, which means we move the start point of the window. The new start point is determined based on the previous start point and the size of the window before it was extended, which we refer to as the *default* window size. Since we do not know exactly where the connection loss is, one option is to set the start point of the new window beyond the end of the previous default window.

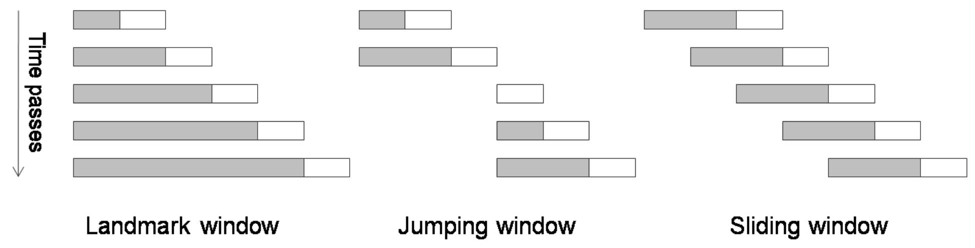

Figure 4.8: Different types of window algorithms [DZone 2010]

Having *enough* samples in the window is determined based on the current size of the window and the sampling rate. The sampling rate gives us the expected number of samples each second, which we multiply with the window size (in seconds) to calculate the expected number of samples for a given window size. A sampling rate of 12 Hz and a window of 10 seconds means we expect: $12\,Hz \times 10\,sec = 120$ samples for this window. However, we need to consider the implications of a change in sampling rate, as the sampling rate of the FLOW sensor is not necessarily stable. If the sampling rate is 10 Hz, but we expect it is 12 Hz, we only get 100 out of the 120 samples we expect for a time window of 10 seconds. We may falsely detect the *missing* samples as a connection loss. The problem is that we cannot know for sure if it is a drop in the sampling rate or a small connection loss because of the jitter and unreliable sampling rate of the FLOW sensor. One option is a visual inspection of the signal which should reveal if there is a gap. This is not a feasible solution for longer recordings as it will be too time-consuming. As a result, we may not find a perfect definition of having *enough* samples in a window to adjust the timestamps correctly while at the same time accurately detecting all connection losses.

Figure 4.9 illustrates how the jumping window should move as time passes. In this example, the red line represents time, and the blue squares represent when a packet of samples from the sensor arrives at the application on the smartphone. The default window of window A, B, and C are the same size and needs three packets to be *filled up*. In other words, three packets are enough to adjust the timestamps correctly for the time period of the default window. This is the case for window A since three packets arrive in the default window. We move the start point of the new window past the default window of A, illustrated with window B in Figure 4.9. For window B, only one packet arrives in the default size. We extend window B with the default window size four times and finally get a total of four packets in the extended window at location t_1, which is enough to fill the default size of window B. However, the current size of window B at t_1 is now four times the size of the default window and only contains four out of the twelve packets needed. We extend window B once again to look for the missing packets and reach the maximum window size at location t_2. Although more packets arrive, window B only contains nine out of fifteen packets needed. The result is the detection of a connection loss since window B cannot be extended further.

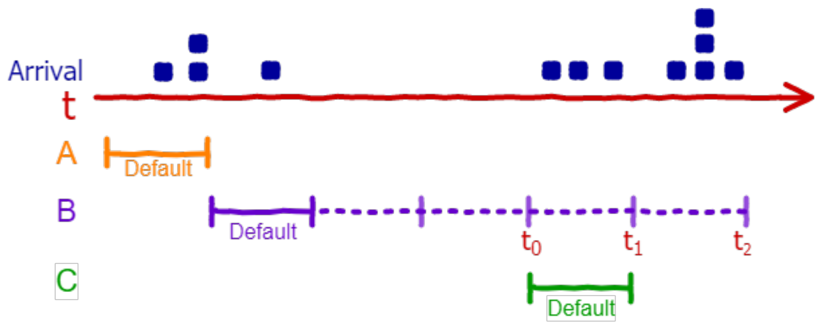

Figure 4.9: Illustration of jumping window

There is one packet in the default size of window B that we need to handle before moving the window past the detected connection loss. We discard the samples in this packet because we cannot adjust the timestamps correctly. The reason is that we do not know if the samples should be distributed evenly in the default window or more towards the start or end-point. The start-point for the new window after a gap is estimated based on the explanation in Section 4.3. This start-point is shown as t_0 in Figure 4.9 and is the start-point for the window C. The default size of window C also contains three packets, which means we can adjust the timestamps without extending the window.

The result of using the jumping window model to adjust the timestamps of FLOW is shown in Figure 4.10. The two subplots show that the signals are synchronized at the beginning of the recording, as seen in Figure 4.10a, and still after seven hours of recording, as seen in Figure 4.10b. Since the recording is eight hours long, it is too time-consuming and tedious to visual inspect and verify that the signals are accurately synchronized at all times. After checking the signals at random times throughout the recording, there is nothing to suggest that the signals are not synchronized.

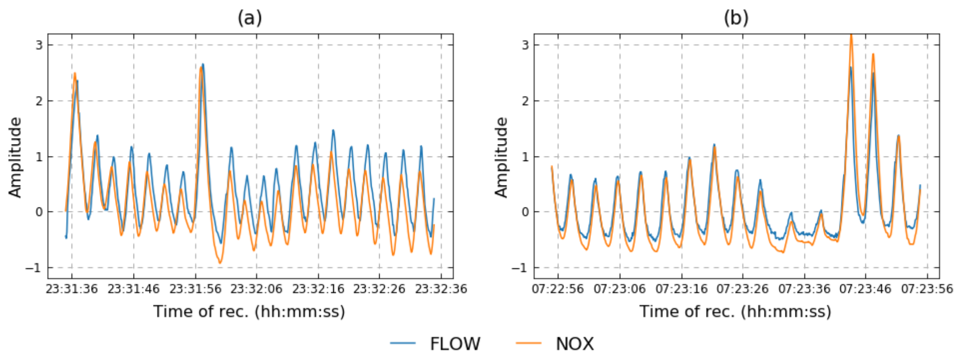

Figure 4.10: Example of synchronized signals using the jumping window

Adjusting the timestamps with the window approach the signals are synchronized after seven hours of recording, as seen in Figure 4.10. Although this is a promising result, it is important to note that it only proves the signals are synchronized in the given period and that the remaining hours of recording are not necessarily synchronized. This is, however, very time-consuming to manually inspect and verify. Another option is to use the breath detection algorithm and compare for each breath the distance between the peaks in the FLOW signal and NOX signal. One problem with this approach is that peaks from motion artifacts can be detected as false breaths. If we compare the distance between a false breath in one of the signal with a real breath in the other signal, the result will be misleading. To avoid this we can compare only the breaths that have been matched in the two signals. However, breaths are matched because the synchronization is accurate enough that the peak of the FLOW breath is inside the start and end of the corresponding NOX breath. In the end, breaths that are not false and are not matched, yields more information about the synchronization accuracy, which is what the sensitivity and positive predictive value metrics represent later on during our evaluation.

Discussion and Conclusion

To summarize, we have presented five design alternatives for timestamp adjustment. The first alternative is based on the results from Løberg [Løberg 2018], where the timestamps in shorter FLOW recordings have successfully been adjusted according to Equation 4.1. Unfortunately, there is a tendency that the sampling rate of FLOW sensors is not stable. This problem might be negligible for shorter recordings, but for longer recordings, a constant sampling rate is important, or else we cannot adjust the timestamps based on a fixed sampling rate. For this reason, the second alternative that estimates the sampling rate is also not sufficient since the sampling rate used is still fixed. A solution to this problem was presented in the third alternative. By measuring the sampling rate and adjusting the timestamps in intervals, we should be able to adjust the timestamps more precisely throughout the recording.

Another problem we have is the detection of connection loss in FLOW. The first three alternatives we presented cannot solve this problem. As a result, we explore ways to detect connection losses in FLOW. Based on our analysis of FLOW, the timestamp of the last sample in each packet should be the most correct. The reason is that the last sample does not have to wait for other samples to be collected before the packet is sent to the smartphone. Assuming that the packet is processed immediately by the application after the smartphone receives it, then the timestamp of the last sample should be near identical to the actual time it was recorded by the sensor. However, this assumption is not true as jitter means we cannot trust every packet to be processed as soon as they arrive on the smartphone. Sometimes multiple packets are processed together, which suggests that packets are queued somewhere on the smartphone.

The final alternative we presented is the jumping window model. The timestamps are adjusted using a landmark window that essentially is a flexible interval. The estimated sampling rate is used to ensure that *enough* samples are present in the window before we adjust the timestamps. Variations in the sampling rate can be handled by distributing the samples evenly in the window when we adjust the timestamps. This effectively means that if there are more samples in the window than required, the samples are adjusted closer together. Determining how many samples are *enough* in a window is challenging because we want a flexible window that can handle sampling rate variations without falsely detecting connection loss.

Of the five design alternative we present, only the last alternative, the window model, is able to adjust the timestamps of longer recordings - while simultaneously detecting connection loss. The results are promising enough that we want to investigate this alternative further. Therefore, we focus on the performance of the jumping window model in the following sections.

4.3 Start Time after Gap

When packets start to arrive after a connection loss, we need to estimate the point in time where the connection between the sensor and smartphone is re-established. Meaning we need to estimate when the first sample in the first packet to arrive after the gap was measured in the sensor. The unreliable time-stamping means that all we know is when the sample was processed by the application (i.e., timestamped). We can, however, estimate the correct timestamp of the first sample to arrive after a gap if we assume that the timestamp of the last sample in the first packet to arrive is correct. This assumption is based on the fact that multiple samples often get identical timestamps as the time it takes to process a sample for the application is minimal. It is possible that multiple packets arrive together after a gap due to jitter. In this case, we always use the timestamp of the last sample in the batch.

Figure 4.11 illustrates which samples are used to estimate the start time after a gap. In this example, each small blue line in the packets represents the timestamp of a sample. After the gap, a batch of three packets arrives together. Each sample in this batch is processed in order, with the sample t_0 first and the sample t_1 last. To locate the last sample t_1 in the batch, we look for a large time difference in timestamps between two subsequent samples. The last packet, as seen in Figure 4.11, arrives much later than the first three packets. The timestamp of ts_2 is therefore significantly higher than ts_1. We have found the last sample in the first batch, as these two samples have not been processed together. To estimate the time of ts_0, we use the sampling rate, the number of samples in the batch, and the timestamp of the last sample in the first batch. The timestamp of t_0 is calculated using Equation

4.3, where N is the number of samples in the first batch, and the sampling rate is converted to the *interval*. Using the example from Figure 4.11 with sixteen samples in the first batch and an interval between each sample of 100 ms (10 Hz), t_0 is estimated as: $t_1 - 16 \times 100\, ms$ or 1.6 seconds before t_1.

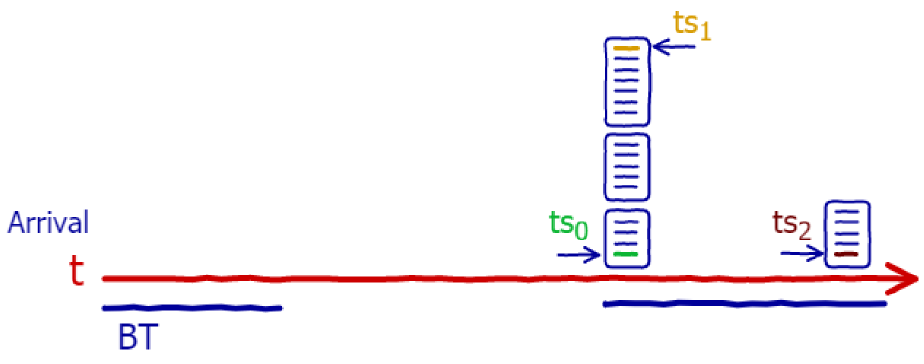

Figure 4.11: Illustration of how the start time after a gap is estimated

$$t_0 = t_1 - (N \times \frac{1000}{Hz}) \tag{4.3}$$

After we have estimated the point in time when the connection is reestablished, we can use it as the new start-point of the window. This approach is also used to estimate the start time of a recording. This method assumes that we have identified a connection loss as opposed to a gap caused by jitter. Severe cases of jitter cannot be handled by using this approach, but the current level of information for FLOW recordings means we cannot perfectly adjust the timestamps in any way. Therefore, we could decide to exclude the first seconds after a gap, but as we see later on, it has no significant impact on the data we have processed.

The function used to estimate the start time of a recording or the timestamp of the first sample after a gap can be seen in Listing 4.1, where *ts* are a list of all samples timestamps. We locate the index of the last sample in the first batch to arrive by comparing the time difference of two subsequent samples. If the time difference is higher or equal to the interval, we have found the last sample in the first batch to arrive. The function returns the estimated timestamp using Equation 4.3.

```python
def estimate_timestamp(ts, interval=100):
    for i in range(len(ts)-1):
        if ts[i+1] - ts[i] >= interval:
            return ts[i] - i * interval
```

Listing 4.1: Function for estimating the start time of a recording or the timestamp of the first sample to arrive after a gap

4.4 Sampling Rate Measurement

The sampling rate of a sensor is the average number of samples it measures each second. Measuring the sampling rate of a recording is trivial as long as no data is missing and we know the duration of the recording. It can be measured using the following equation: $f_s = \frac{samples}{duration_s}$. The duration is easily calculated based on the start and end time of the recording. Slightly wrong estimation of the start time only matters when the duration of the recording is short. However, larger periods of connection loss have a great impact, because the number of samples decreases while the duration stays the same. As a result, the calculated sampling rate is lower than the actual sampling rate.

To estimate the sampling rate as precise as possible we want to exclude periods of connection loss from the measurement. However, the problem of whether a gap is caused by jitter or a loss of connection still exists. One way of estimating the sampling rate is to calculate the average sampling rate over a time interval. With this approach, it is possible to exclude periods without any samples when calculating the average sampling rate. Measuring the sampling rate in short periods means that the sampling rate is measured many times and that a few periods of zero samples will not greatly impact the average calculated sampling rate. For example, if we calculate the sampling rate a thousand times to be exactly 10 Hz and once to 0 Hz, then the average sampling rate is still 9.99 Hz. By measuring the sampling rate multiple times with different intervals, we should end up with a decent estimation of the actual sampling rate for a recording. In any case, it does not have to be exactly precise, since the window model should keep the signals synchronized as long as the sampling rate is somewhat estimated correctly.

The function used for measuring the sampling rate can be seen in Listing 4.2, where the interval used for measuring the sampling rate is given in seconds. This function also takes as argument a *pandas* data frame, which is a mutable two-dimensional data structure consisting of rows and columns similar to tabular data. The last option is to exclude the intervals without any samples from the list used to estimate the average sampling rate of the recording. It is important to note that this function estimates the overall sampling and does not consider underlying changes in the actual sampling rate.

```
def estimate_sampling_rate(dataframe, interval=60, exclude=False):
    ts = datetime.timedelta(seconds=interval)
    time = dataframe.index[0]
    sampling_rates = []
    while time < dataframe.index[-1]:
        samples = len(dataframe.loc[time:time+ts])
        time += ts
        if samples == 0 and exclude:
            continue
        sampling_rates.append(samples / interval)
    return numpy.mean(sampling_rates)
```

Listing 4.2: Function for measuring sampling rate

4.5 Preliminary Testing

The general purpose of our preliminary testing is to determine if the window model detects connection losses correctly and how accurate the window model adjust the timestamps. By comparing the signal quality metrics of different window sizes, we should be able to determine if there is an optimal size to use for the default window. The main hypothesis is that a small default window should adjust the timestamps of FLOW more accurately than a larger default window. Another important aspect is how precisely we can determine where a connection loss begins. Additionally, these preliminary tests may uncover sensor-specific oddities or other issues that should be taken into account during preprocessing.

To perform the testing of the window model, we use a thirty-minute segment from one of the FLOW. recordings. We have chosen to use a segment that is definitively missing data because of a connection loss. The segment is shown in Figure 4.12, where the connection loss lasting for around two minutes, can be seen in the middle of the recording. When the connection re-establishes, the baseline of FLOW is shifted. This segment should enable us to test the accuracy of different default window sizes and their ability to detect connection loss.

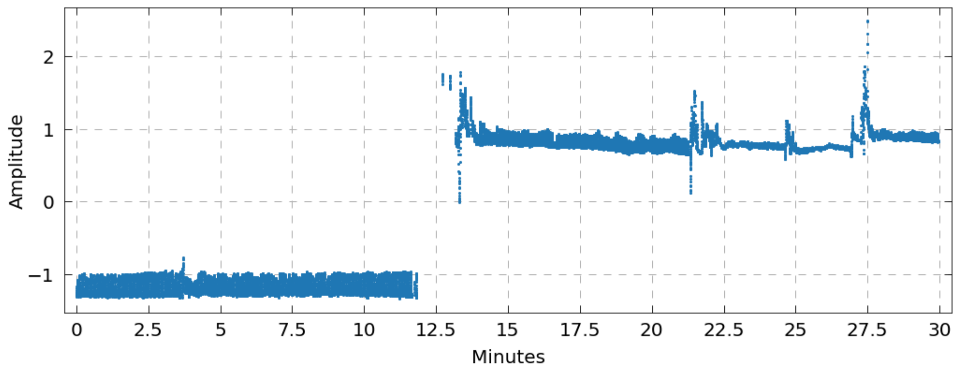

Figure 4.12: Thirty-minute segment from a FLOW recording

4.5.1 Findings

Default Window Size

During the timestamp adjustment of the thirty-minute segment, the first notable thing we discovered is that both a small and a larger default window size struggle with adjusting all of the timestamps correctly. A small selection of the segment with the adjusted FLOW timestamps using a one-second and thirty-second default window is shown in Figure 4.13a and Figure 4.13b. We want the FLOW signal to *align* in time (synchronize) with the NOX signal as accurately as possible. Although most of the FLOW signal in Figure 4.13a is synchronized with the NOX signal, they are less synchronized around the twenty-second mark. The exact reason remains unknown, but it only affects a few breaths before the signals are synchronized again. This is not the case for the FLOW signal in Figure 4.13b. The signal is not synchronized with NOX, evident as almost no corresponding breaths in the two signals are aligned in time. It is only towards the end of this period that the signals realign, which means that a larger period of the timestamps are inaccurately adjusted compared to Figure 4.13a. This indicates that using a small default window is better for accurately adjusting the timestamps of FLOW.

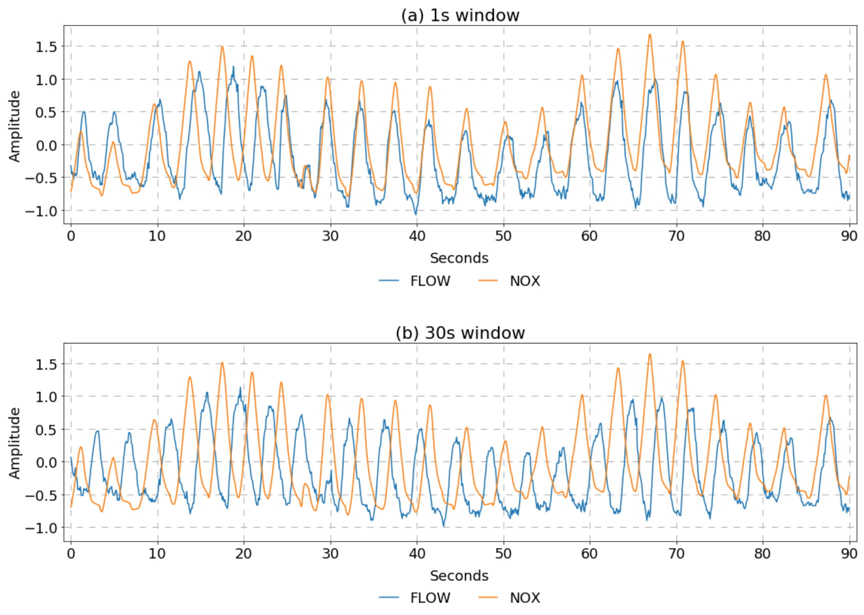

Figure 4.13: Example of synchronization precision of different default window sizes

We use the signal quality metrics to compare how accurate the different default window sizes are at adjusting the timestamps of FLOW. A high sensitivity, PPV, and CMP score means that breaths in the two signals are aligned/synchronized. The signal quality metrics of different default window size on the thirty-minute segment is shown in Table 4.5.

One may first notice that the sensitivity and PPV scores are very similar across the different default window sizes. If we instead consider the CMP score, the three smallest default windows of 0.7s, 1.0s, and 1.4s achieve the highest score of 96.0%. A thirty-minute segment means that each minute accounts for roughly 3.33% of the final score, although this might deviate slightly based on how the signals are synchronized. A score of 96.0% means that approximately twenty-nine out of the thirty minutes are clean.

Default window	Sensitivity	PPV	CMP	WAPE
0.7 sec	100.0%	99.77%	96.0%	4.33%
1.0 sec	100.0%	99.77%	96.0%	4.47%
1.4 sec	100.0%	99.77%	96.60%	4.30%
2.1 sec	99.77%	99.53%	89.86%	4.38%
2.8 sec	99.77%	99.77%	93.29%	4.51%
3.5 sec	100.0%	99.53%	93.19%	4.40 %
7.0 sec	97.97%	98.39%	83.30%	4.47%
30.0 sec	98.18%	98.84%	79.95%	5.23%
60.0 sec	99.09%	99.53%	89.91%	5.36%

Table 4.5: Quality metrics for the thirty-minute segment using different default window sizes

The lowest CMP score of 79.95% for the thirty-second default window reflects the observation from Figure 4.13 that larger windows appear to adjust timestamps less accurately. A CMP score of 79.95% means that approximately six of the thirty minutes are dirty with either a sensitivity or PPV score of less than 100%, which indicates that the signals are not perfectly synchronized.

Detecting Connection Loss

Detecting connection loss in the FLOW signal is essential in order to adjust the timestamps correctly. As explained in Section 4.2.5, when a connection loss is detected, we discard any samples in the default window leading up to the connection loss as we cannot adjust the timestamps of these samples correctly, or else the window model would not have been extended in the first place. During testing, we discovered that the default window size also affects how *many* samples we discard before a connection loss.

Figure 4.14a and Figure 4.14b shows the NOX signal and the FLOW signal where the timestamps have been adjusted using a 700ms or a 60sec window. The two NOX signals are aligned in time, which means the different appearance of the FLOW signals is because of our timestamp adjustment. The connection loss in FLOW from the thirty-minute segment is noticeable in the middle of both figures, when only the NOX signal is shown. One may also notice that the start and end time of the connection loss is different between the FLOW signals, although their duration is similar. In Figure 4.14b, the FLOW signal is lost more than half a minute before compared to Figure 4.14a but also re-establishes the connection faster. However, the FLOW signal in Figure 4.14b is no longer synchronized with the NOX signal after the gap, but the FLOW signal in Figure 4.14a is. The first minute after the gap is noticeable noisier for both FLOW and NOX, which may be the reason that the larger default window of 60sec is less accurate at adjusting the timestamps compared to the small default window of 700ms.

Our findings indicate that a large default window of 60sec may remove/discard more samples before a connection loss compared to a small window of 700ms. We may choose to exclude the first minute or so after a connection loss as this signal period is noisier and may not be accurately synchronized as evident when adjusting the FLOW timestamps with a large default window. It is interesting that NOX is recording normally during the period of connection loss in FLOW. The connection re-establishes during a noisy period (possible movement of the patient), which can indicate that the FLOW signal was blocked.

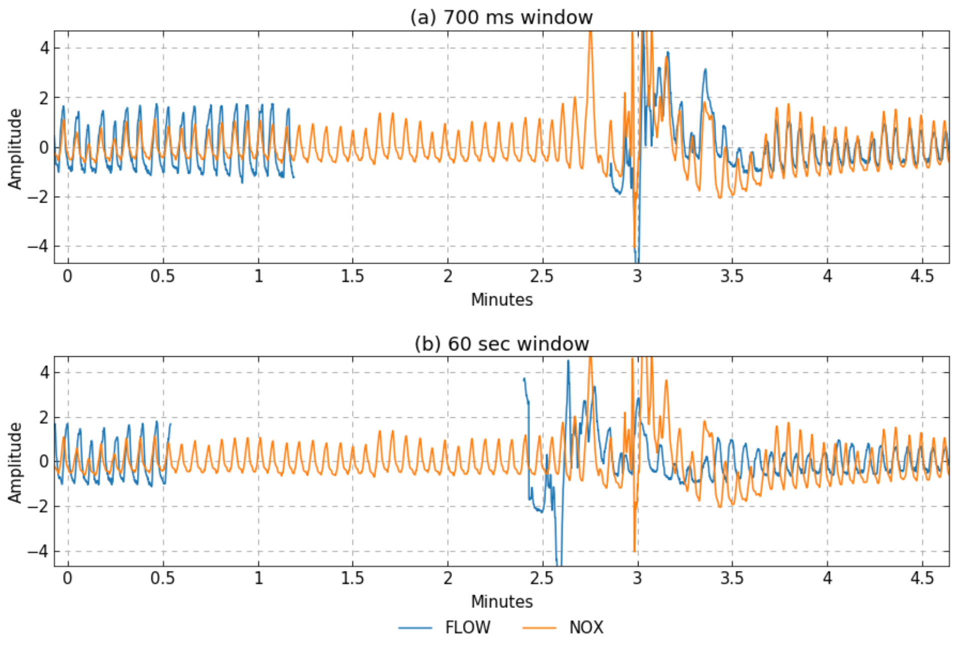

Figure 4.14: Example of connection loss detection with different default window sizes

Signal Quality Metrics

The requirement for using the breath detection accuracy metrics (sensitivity, PPV and CMP) and the breath amplitude accuracy metric (WAPE) to evaluate the FLOW recordings is that the FLOW signal is synchronized with the NOX signal. The synchronization does not have to be 100% exact for the metrics we are calculating. The only requirement is that the peaks of the real breaths in the signal from FLOW are aligned, such that they are located somewhere between the start and end of the corresponding breaths in the signal from NOX. We want to test how the accuracy of our synchronization correlates to the results of the signal quality metrics. The idea is that if our timestamp adjustment for some parts of a FLOW recording is inaccurate, then the result of the signal quality metrics should reflect that FLOW and NOX are not synchronized in these specific parts of the recording.

To test this, we use a period from a recording where we have manually synchronized the FLOW and NOX signals. We then shift the FLOW signal in time, which means that the timestamp of every sample in the signal is shifted. The results of measuring the signal quality metrics with the shifted FLOW signal are shown in Table 4.6. Shifting the FLOW signal one second out of sync lowers the sensitivity and PPV scores with around 2% compared to the synchronized FLOW signal. This means that most breaths are still correctly detected. However, the low CMP score of 46.67% means the errors of false breaths or undetected breaths, are evenly distributed throughout the FLOW signal.

Signal	**Sensitivity**	**PPV**	**CMP**	**WAPE**
Synchronized	99.10%	98.65%	73.33%	5.95%
Shifted 1 sec	97.27%	96.83%	46.67%	7.68%
Shifted 2 sec	82.65%	81.90%	0.00%	30.44%
Shifted 3 sec	88.64%	88.24%	6.67%	51.75%
Shifted 4 sec	95.91%	95.91%	40.00%	51.46%

Table 4.6: Example of how the synchronization accuracy affects the results of the signal quality metrics

Except for the WAPE score, it is interesting that the signal quality results when shifting the FLOW signal four seconds is better than when shifting the signal two seconds. The reason is that we have shifted the FLOW signal far enough that many non-identical breaths in the two signals align. An example of this can be seen in Figure 4.15, where none of the breaths that are aligned in time are alike. It is only the results of the WAPE metric that is significantly different between the FLOW signal that is slightly out of sync (shifted 1s) and the FLOW signal that is more out of sync (shifted 4s). In the FLOW signal shifted four seconds, the WAPE score around 51% indicates that there is no clear relationship between the breath amplitudes in the two signals, which makes sense as the breaths compared are not identical.

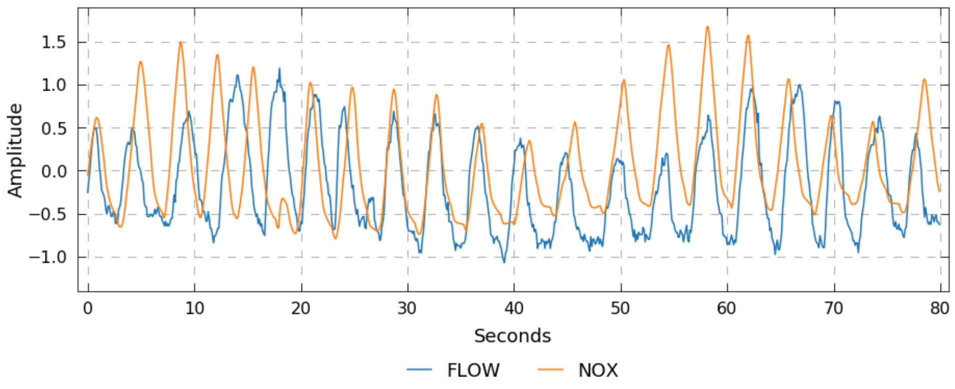

Figure 4.15: Example of non-identical breaths aligning in time

Sensor Oddities

During testing, we discovered that both the signal from FLOW and NOX are noisy at the beginning of a recording. We suspect that this may be caused by sensor calibration and movement by the subject. An example of sensor initialization can be seen in Figure 4.16. Notice that the signal from NOX flat-lines multiple times as indicated by a continuous straight signal line. The missing signal period in the middle of the figured is caused by a connection loss in the signal from FLOW. During synchronization, we also remove the NOX signal in the period corresponding to the connection loss in the FLOW signal. It is normal for people to change sleeping position several times before they fall asleep, thus, we may choose to remove the first several minutes of a recording while the signals calibrate, and the patient falls to sleep.

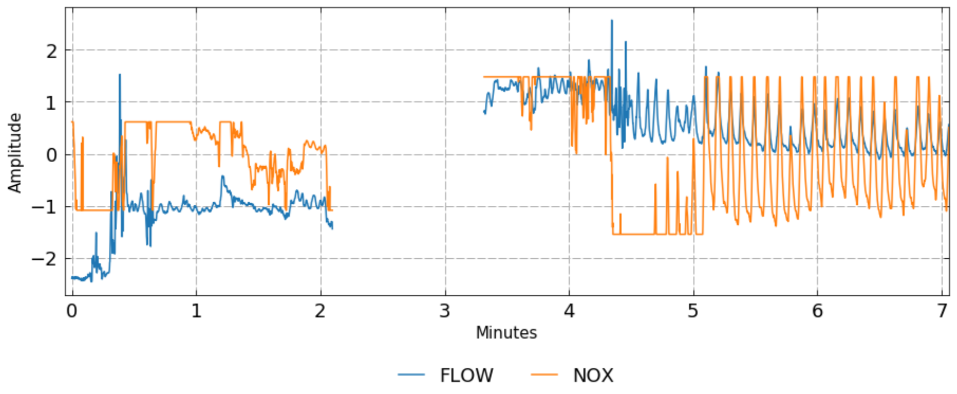

Figure 4.16: First seven minutes of an overnight recording

Another oddity we discovered is that the amplitude peak of breaths in the signal from NOX is sometimes cut off, as seen in the example in Figure 4.17. See, for example, the breaths around the one- and two-minute mark, where several breaths are limited to the same maximum peak amplitude. One possible explanation is that this is an unfortunate effect of the automatic signal preprocessing performed by the Noxturnal software system. Although this oddity only exists in a minor part of the signal, it will nevertheless be misleading when we measure the breath amplitude accuracy.

4.5.2 Discussion

Our preliminary testing indicates that using a small default window is a better option for adjusting the timestamps of FLOW. Choosing the maximum size the window can extend to, depends on how severe the jitter is. To allow for a flexible solution, that can be adapted to the preprocessing needs of specific recordings the default window size, the maximum size it can extend to and what we consider to be *enough* samples in a window, should be parameters of the window model that can easily be modified.

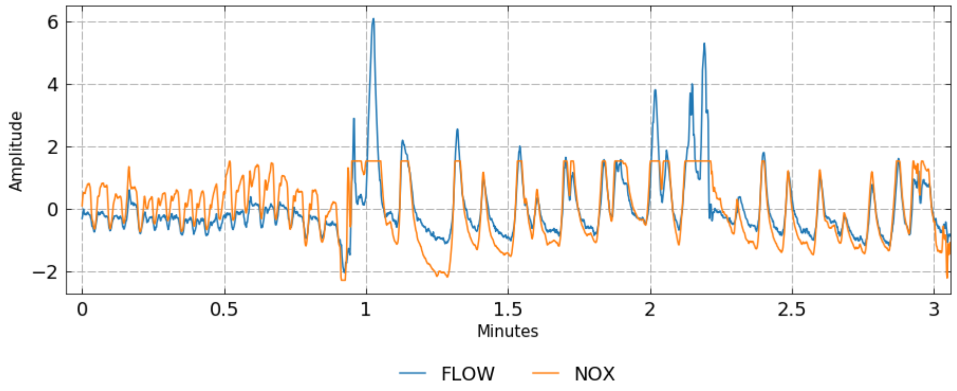

Figure 4.17: Limited amplitude peak of breaths for NOX

4.6 System Environment

We have chosen Python as the programming language to implement the scripts in, mainly because we modify and reuse existing Python scripts. There are libraries available for Python that contain some of the needed functions, such as *pyEDFlib* [pyEDFlib 2018] for processing European data format (EDF) files, *pandas* [pandas 2018] for CSV-parsing and data manipulation, and *NumPy* [numPy 2020] for timestamp adjustment. All of these libraries contain the functions used by the existing scripts for resampling, interpolation, synchronization, and regression. To visualize the signals, we use the *matplotlib* [matplotlib 2018] library to plot the data in a graph. At last, the existing script used to measure the signal quality uses a *findpeaks* function available in MATLAB [The Mathworks, Inc. 2018a]. A MATLAB Python API is used to call the functions directly from Python as if it were a normal library. MATLAB and the API must be installed on the system. The software and versions we use during preprocessing and signal quality evaluation are listed in Table 4.7.

Software	**Version**
pandas	0.23.4
Python	3.6.8
pyEDFlib	0.1.14
NumPy	1.16.4
SciPy	0.19.1
scikit-learn	0.19.0
matplotlib	2.0.2
MATLAB	R2018a
MacOS	10.14.6

Table 4.7: Software versions used during preprocessing and signal quality evaluation

pandas [pandas 2018] is an open-source library for data manipulation and analysis. It contains data structures, tools, and functions for parsing of CSV, resampling, and interpolation. Sliding window functions are available, but they cannot be customized to suit our needs for a jumping window method.

pyEDFlib [pyEDFlib 2018] is an open-source library to read and write EDF files in Python built on the EDFlib library written in C language. We have made some changes to the source code because of some errors that occur when reading EDF files. The errors occur because we do not include all of the signals when exporting the recording from the Noxturnal Software Program in order to save space.

NumPy [numPy 2020] is a well renowned open-source library useful for scientific computing. It supports functions for handling large, multi-dimensional arrays and matrices, and mathematical operations. In our implementation we make use of functions such as *linspace* to distribute timestamps evenly.

SciPy [SciPy 2017] is an open-source library that includes signal processing capabilities. The preprocessing script uses the cross-correlation function during the synchronization of signals.

scikit-learn [scikit-learn 2018] is an open-source library useful for data analysis and machine learning. The *TheilSen* linear regressor is used during the signal quality measurement.

matplotlib [matplotlib 2018] is an open-source library well-documented for 2D plotting in Python. We use the library for creating box-plot figures and for plotting signals in a line chart.

MATLAB [The Mathworks, Inc. 2018b] is a commercial platform consisting of a multi-paradigm numerical computing environment, and it includes its own proprietary programming language. A license to the platform is available through the University of Oslo (UiO). We only use MATLAB during the measurement of signal quality.

4.7 Preprocessing

The complete dataset needs to be preprocessed before we can measure the signal quality of overnight FLOW recordings. In addition to timestamp adjustment and connection loss detection, we need to extract the abdomen signal from NOX recordings. The signal from FLOW and NOX will both be converted to a CSV-file to unify the data formats. We use a modified version of a script to synchronize and combine the signals. We present the implementation of three scripts we need to preprocess our dataset in this section, with the following tasks:

1. Preprocessing of FLOW, including timestamp adjustment, connection

loss detection, and date format conversion.

2. Extraction of the abdomen signal from NOX T3, including timestamp generation.

3. Combining and synchronizing non-connection loss periods in FLOW with NOX. This includes resampling, interpolation and combining all parts into one file.

4.7.1 FLOW Preprocessing

A FLOW recording exported from the RawDataMonitor application consists of a folder with multiple text files (.txt). In this folder, there is one *breathing.txt* file that contains the raw breathing data for this particular recording. An example of a *breathing.txt* file can be seen in Listing 4.3. Each row consists of one entry with multiple values separated by a single space. We are only interested in two columns; the third column that measures the current stretch of the strain-gauge belt (highlighted in blue) and the corresponding UNIX timestamp in column eight (highlighted in orange). The UNIX timestamps are listed with a precision of one millisecond.

```
0 0 1603 0.000000 11 0.000000 0.000000 1535917933038 true
0 0 1601 0.000000 11 0.000000 0.000000 1535917933040 true
0 0 1604 0.000000 11 0.000000 0.000000 1535917933041 true
0 0 1606 0.000000 11 0.000000 0.000000 1535917933042 true
0 0 1603 0.000000 11 0.000000 0.000000 1535917933043 true
0 0 1602 0.000000 11 0.000000 0.000000 1535917933044 true
```

Listing 4.3: Example of breathing.txt output file from SweetZpot's sleep recording application

The main part of the script we use to preprocess FLOW can be seen in listing 4.4. Since the script is rather long, we have omitted the argument parsing for clarity. The script utilizes the *pandas* library to easily parse text files, manipulate grouped data, and for DateTime conversion. To adjust the timestamps, we use the *linspace* function included in the *NumPy* library, which returns the samples evenly spread out over an interval. This script only requires the breathing.txt file as argument to run as the remaining arguments are optional. This includes the default window size, the maximum window limit, and the minimum allowed sampling rate in a window in order to adjust the timestamps. For instance, a minimum sampling rate of 9 Hz with a window size of 1 second means we decide that 9 samples are *enough* to adjust the timestamps.

The values of the optional arguments are preset based on the results of preliminary testing, but can be altered to suit different needs. Additionally, the *estimate_rate* argument should be used if one wants to use the estimated sampling rate of the recording to determine the minimum sampling rate allowed in a window. The script will write a comma-separated values (CSV) file to *standard output*, which can be redirected to a file if desired.

The following is an example of how to run the script:

```
$ python flow-cleaning.py --file=breathing.txt --default=0.7
    --limit=15 --min_rate=9.0 > flow.csv
```

In this example, most arguments are specified. The `--default` argument specifies that the default window size should be set to 700 milliseconds, which is also the size the window increases in each iteration. If there are insufficient samples in the default window, the window is extended with the default window size in each iteration until it reaches the maximum window limit. The `--limit` argument specifies that the maximum window extension is 15 seconds. The `--min_rate` argument specifies the minimum sampling rate allowed to adjust timestamps in the window. The output of this script is redirected to new a CSV file stored as *flow.csv*, which consists of two columns; the adjusted timestamps formatted to a more readable datetime format, and the raw signal.

The script works by continuously updating the window in a loop as long as the end-point of the current window is in the time-range of the recording. In each iteration of the window, three scenarios are possible. If enough samples are included in the current size of the window, then we adjust the timestamps of the samples, or else, if the window has reached the limit, we detect a connection loss. If none of the above is true, then we extend the window and run the same checks again. The script estimates the start time of the recording and the timestamps of the first sample after each gap, based on the explanation given in Section 4.3.

```python
# Parse the arguments into a dictionary
args = parse_arguments(sys.argv)

# Read breathing.txt file
signal = pandas.read_csv(args['file'], sep=" ", header=None, usecols=[2, 7], names=[1,0])

# Est. start time of recording
signal[0].values[0] = estimate_time(signal[0].values)

# Use est. sampling rate as minimum rate allowed if specified
if 'est_rate' in args:
    args['min_rate'] = estimate_sampling_rate(signal.index)

# Use timestamp column as index
signal.index = signal[0]

# Align window
t_from = t_to = signal.index[0]

while t_to < signal.index[-1]:
    t_to += args['default'] # Increase end point of window

    samples = len(signal.loc[t_from:t_to])
    rate = samples / (t_to - t_from)

    # Sampling rate in window satisfies minimum sampling rate
    if rate >= args['min_rate']:
        signal.loc[t_from:t_to, 0] = adjust_timestamps(t_from, t_to, samples)
        t_from = t_to # Move window further
        continue

    # Connection loss. Move window past default window and estimate start time after gap
    if t_to - t_from > args['window_limit']:
        t_to = estimate_time(signal.loc[signal.index >= t_from + args['default'], 0].values)
        signal = signal.drop(signal.loc[t_from:t_to].index)
        signal.index = signal[0] # Reset index after drop
        t_from = t_to

# Adjust timestamp for samples in last window if any
samples = len(signal.loc[t_from:t_to])
if samples != 0:
    signal.loc[t_from:t_to,0] = adjust_timestamps(t_from, t_from + samples * 100, samples)

# Drop raw timestamps
signal = signal.drop(columns=[0])

# Convert UNIX timestamp to local datetime
signal.index = pandas.to_datetime(signal.index, unit="ms", utc=True).tz_convert('Europe/Oslo')

signal.to_csv(sys.stdout, header=False)
```

Listing 4.4: Script for preprocessing FLOW

4.7.2 NOX Preprocessing

The Noxturnal software program facilitates the storage of Nox T3 recordings. The export of data to CSV file format from Noxturnal is restricted to 1 Hz, even though the actual sampling may be higher. For instance, the actual sampling rate of the RIP abdomen signal is 20 Hz. Data from Noxturnal can also be exported to EDF, which maintains the original sampling rate of the signals. Since the sampling rate of FLOW is approximately 10 Hz, we have chosen to export the NOX signal to EDF to maintain a higher data granularity when synchronizing the signals from the two sensors. This means that another preprocessing step is required in order to convert the NOX signal data from EDF to CSV.

EDF was originally designed for storage and exchange of multichannel biological and physical signals [Alvarez-Estevez, Diego 2020]. The simple and flexible format of EDF means it is commonly used for storing commercial data independently of the acquisition software. According to the specification of EDF, one EDF file can only contain one uninterrupted recording. In our dataset, each patient recorded their sleep with NOX T3 for two consecutive nights. As a result, each EDF file we export consists of two overnight recordings, as shown in Figure 4.18. In this figure, we have extracted the NOX T3 RIP abdomen signal from the EDF file. Notice that around eight hours of recording, the NOX signal flatline which indicates that the patient was no longer wearing the NOX belt. The signal starts recording again around the following evening or night, that is, after twenty-four hours.

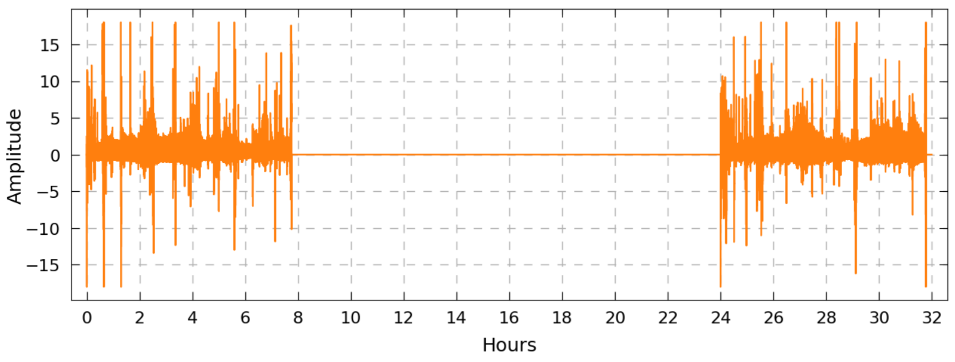

Figure 4.18: Uninterrupted EDF file with two overnight recordings

To convert EDF to the CSV format, we utilize the pyEDFlib library specifically created to read and write EDF files in Python. By using this library, we can extract individual signals, such as the respiratory effort signal from the abdomen or thorax. EDF files contain a header with information such as the start time of the recording, duration, signal label (abdomen, thorax, etc.), and the sampling rate for each signal. During the conversion to CSV, we have to timestamp every sample ourselves as this information is not directly stored. This is, however, an easy task to perform, since the start time and sampling rate information is stored in the header file.

We discovered during the exportation to EDF from Noxturnal that some of the recordings have an incorrect start time. These particular screenings were conducted right after the daylight saving time ends in central Europe, which means that the timezone is off with one hour compared to FLOW. To correct the timezone of these particular recordings we utilize the *tz_localize* function from *pandas*.

The main part of the script we use to extract the NOX abdomen signal from EDF can be seen in Listing 4.5. By default, the script exports and converts both nights of recordings into one CSV file. This is not a problem since the preprocessing script we use to synchronize the FLOW and NOX signals compares the timestamps and extracts the time-period that matches both signals. Several arguments can be passed to this script. Exporting one night of recording or a part of the night can be achieved using the *start* and *end* arguments. This is useful since the complete exported CSV file of two nights is around 100 Megabytes in size. The file-size is so large because the sampling rate of the abdomen signal is 20 Hz, and the recording time is often 32 hours long, meaning there are two nights plus the day in between. Further, we have also included the option to *resample* the NOX signal as the signal will be downsampled to 10 Hz later anyway when combining it with FLOW.

The only required argument to run this script is the NOX EDF file to convert. The output of the script is a new CSV file containing the abdomen signal with the generated timestamps. The output is written to *standard output*, which can easily be redirected to a new file if desired. Additionally, this script accepts arguments for *start/end* time, and *resampling*. The optional *start/end* arguments specify in hours what part of the signal to include in the output. The *resample* argument specifies the sampling rate to output. The *localize* argument is used to adjust the timezone to local time when needed.

The following is an example of how to run this script:

```
$ python edf-to-csv.py --file=nox.edf --start=0
    --end=5 --localize --resample=10.0 --show > nox.csv
```

In this example, all arguments are specified. The `--start=0` and `--end=5` arguments specifies that the first five hours of recording should be extracted. The `--resample=10.0` argument specifies that the signal should be resampled to 10 Hz, and the *localize* argument adjusts the timestamps to local time in Oslo. The result is a CSV-file stored as *nox.csv* with two columns; the timestamps and the abdomen signal.

```python
# Parse the arguments into a dictionary
args = parse_arguments(sys.argv)

if 'file' not in args:
    print_usage(sys.argv[0])
    exit(1)

# Read EDF-file
edf = pyedflib.EdfReader(args['file'])

for i, label in enumerate(edf.getSignalLabels()):
    if label.lower() == 'abdomen':
        signal = edf.readSignal(i)
        freq = 1000 // edf.getSampleFrequency(i)
        freq = datetime.timedelta(milliseconds=freq)
        break

time = edf.getStartdatetime()
timestamps = []

# Generate timestamps for samples
for _ in range(len(signal)):
    timestamps.append(time)
    time += freq

df = pandas.DataFrame({1 : signal}, index=timestamps)

if 'localize' in args:
    df.index = pandas.to_datetime(df.index,
        format='%Y-%m-%d %H:%M:%S.%f').tz_localize('Europe/Oslo')
else:
    df.index = pandas.to_datetime(df.index,
        format='%Y-%m-%d %H:%M:%S.%f')

# Extract part if specified
start = df.index[0]
if 'start' in args: start += args['start']

end = df.index[0] + args['end'] if 'end' in args else df.index[-1]

df = df.loc[start:end]

if 'resample' in args:
    fs_interval = str(1000 // args['resample']) + 'ms'
    df = df.resample(fs_interval).mean()

df.to_csv(sys.stdout, header=False)

if 'show' in args:
    pyplot.plot(df, label="NOX")
    pyplot.legend()
    pyplot.show()
```

Listing 4.5: Script for extracting and converting NOX signal from EDF to CSV

4.7.3 Preprocessing of FLOW and NOX

As previously explained, a script to synchronize and combine signals from respiratory effort sensors have already been implemented in [Løberg 2018]. We utilize the script and modify it to suit our needs. A large proportion of the original code is stripped away for simplicity as we do not need to support other types of signals.

One of the changes to the existing code involves the implementation of a function that splits the FLOW signal at connection loss. Another change from the original code is that we synchronize each FLOW signal period with connection loss with the corresponding NOX period. It is easiest to synchronize two signals that have the same sampling rate. We resample both signals to 10 Hz, since this is the expected sampling rate for FLOW. To resample, we take the *mean* value for samples that end up between existing samples. In cases where the mean value is not available, for example at signal boundaries, we interpolate the missing samples using quadratic interpolation.

The main part of the script can be seen in Listing 4.6. The required arguments for this script are the two CSV files, *flow* and *nox*, to preprocess. Optional arguments for *exclude* and *delay* can be specified, in addition to the *show* argument that plots each of the combined parts. The *exclude* argument specifies the number of samples to exclude at the beginning of the recording and after each connection loss. Removing the first minutes in the beginning or after a connection loss can be beneficial as these periods are often noisy. The *delay* argument overrides the automatic cross-correlation synchronization and is useful for cases where we have to manually synchronize the signals. This argument is an integer, which represents the number of samples, which can be negative, that the FLOW signal should be delayed. The output of this script is a new CSV file consisting of the two signals that have been combined and synchronized. As before, this file is written to *standard output*, and can be redirected to a new file. The optional *show* argument plots the processed signals in a chart for each part of the recording that is synchronized. This is useful for visually verifying that the signals are synchronized.

To give an example of how to run this script:

```
$ python preprocessing.py --flow=flow.csv --nox=nox.csv
    --exclude=600 --show > preprocessed.csv
```

In this example, most of the arguments are specified. The `--exclude=600` specifies that the first 600 samples, for each part we combine, should be excluded in the output. The results of the preprocessing script is stored in a new CSV-file as *preprocessed.csv*.

```python
# Function to parse the arguments into a Python dictionary
args = parse_arguments(sys.argv)

# Read CSV-files
flow = pandas.read_csv(args['flow'], header=None, index_col=0,
    parse_dates=[0], date_parser=parse_timestamp)

nox = pandas.read_csv(args['nox'], header=None, index_col=0, usecols
    =[0, 1],
    parse_dates=[0], date_parser=parse_timestamp)

# Split FLOW at connection loss
flow = split(flow)

# Split NOX to equalize part(s)
nox = [cutLengthOf(nox, to=part) for part in flow]

preprocessed = []

# Iterate over each signal period without connection loss
for sig1, sig2 in zip(flow, nox):
    # Resample
    sig1 = sig1.resample("100ms").mean()
    sig2 = sig2.resample("100ms").mean()

    # Interpolate
    sig1 = sig1.interpolate(method='quadratic')
    sig2 = sig2.interpolate(method='quadratic')

    if 'exclude' in args:
        sig1 = sig1[args['exclude']:]

    # Synchronize
    delay = findDelay(detrend(sig1[1]), detrend(sig2[1]))
    sig1 = sig1.shift(delay)

    # Drop NaNs
    sig1 = sig1.dropna()
    sig2 = sig2.dropna()

    # Equalize lengths
    sig1 = cutLengthOf(sig1, to=sig2)
    sig2 = cutLengthOf(sig2, to=sig1)

    part = sig1 * 1 # make copy
    part[2] = sig2[1] # add column
    preprocessed.append(part)

# Combine each synchronized period into one dataframe
combined = pandas.concat(preprocessed, axis=0)
combined.to_csv(sys.stdout, header=False)

if 'show' in args:
    for part in preprocessed:
        pyplot.plot(standardize(part[1]), label="FLOW")
        pyplot.plot(standardize(part[2]), label="NOX")
        pyplot.legend()
        pyplot.show()
```

Listing 4.6: Script to synchronize and combine FLOW and NOX signals

4.8 Discussion and Conclusion

To summarize, we have investigated several alternatives for adjusting the timestamps and detecting connection loss in FLOW recordings. For most alternatives, the main issue is keeping FLOW synchronized with NOX for extended periods of time. The presence of jitter in recordings makes it difficult to be certain whether a small gap is caused by a loss of connection or if data is buffered somewhere on the smartphone or sensor. However, the jumping window model can adjust the timestamps for extended periods and is the only alternative that also detects connection losses, although jitter means some uncertainty always will exist.

Through preliminary testing, we have discovered that the size of the default window affects the performance of the window model for accurate timestamp adjustment. Jitter and the changing sampling rate of the FLOW sensor means it is difficult to determine the *right parameters* the maximum window size and the minimum sampling rate in a window needed to accurately adjust the timestamps, while at the same time detecting connection losses. Even though we have seen indications that a small window exceeds the performance of a large one, this is not necessarily true for every recording. Therefore, the FLOW cleaning script includes the option to set the window parameters for *default* and *maximum* window size and the minimum *sampling rate* required in a window to adjust the timestamps. These parameters allow for fine-tuning the window model as desired.

For preprocessing the dataset we have implemented a total of three Python scripts, including the existing script in [Løberg 2018] that we have modified to fit our needs. Preprocessing the dataset involves the following steps:

- Adjusting the timestamps of FLOW, including identifying connection loss and converting the data to a CSV file using the *flow-cleaning.py* script.
- Then, extracting the NOX abdomen signal from EDF-files and converting the data to a CSV file using the *edf-to-csv.py* script.
- Finally combining the two signals, using the *preprocessing.py* script.

Execution of the preprocessing procedure may be as follows:

```
$ python flow-cleaning.py --file=breathing.txt
    --default=0.7 --limit=15 --min_rate=9.0 > flow.csv

$ python edf-to-csv.py --file=nox.edf --start=0
    --end=9 --localize --resample=10.0 --show > nox.csv

$ python preprocessing.py --flow=flow.csv --nox=nox.csv
    --exclude=600 --show > preprocessed.csv
```

Chapter 5

Evaluation

In this chapter, we evaluate the usefulness of FLOW for sleep apnea detection. We begin in Section 5.1, with an overview of the clinical dataset. We continue in Section 5.2, with an evaluation of the results of preprocessing data from FLOW and the window models' ability to adjust timestamps correctly and detect connection losses. Next, we evaluate the signal quality results of FLOW and compare it with related works in Section 5.3. In Section 5.4, we present some early results of eight data mining classifiers and their ability to detect disrupted breathing in the signals from FLOW and NOX. Finally, we conclude this chapter in Section 5.5. As a side note, this chapter incorporates parts from related works for measuring the signal quality and classification, with the general flow of data shown in Figure 5.1.

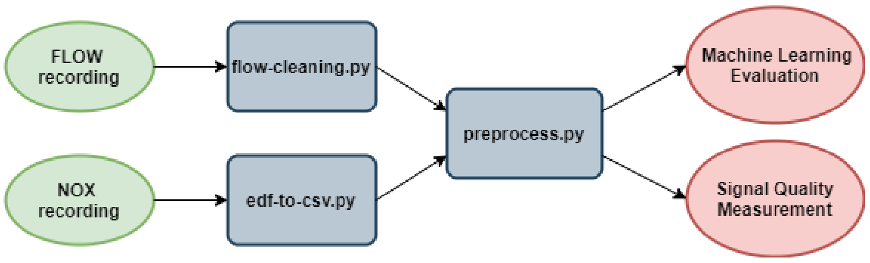

Figure 5.1: Data flow during our quality evaluation

5.1 Dataset

Figure 5.2 presents an overview of our dataset of FLOW and NOX recordings. Figure 5.2a lists the total number of hours recorded with the FLOW and NOX signal together with how much we have been able to synchronize and use for our evaluation. Notice that FLOW has recorded for a significantly lower amount of hours compared to NOX (316 versus 493 hours). The main reason for this is because several FLOW recordings are missing, possibly because the patient forgot to start the registration. Additionally, we have discovered at-least thirteen recordings where FLOW stops recording soon after beginning or in the middle of the night. Furthermore, we have identified approximately twenty hours of missing data during our preprocessing of FLOW recordings, which is due to the intermediate connection losses of the signal while recording.

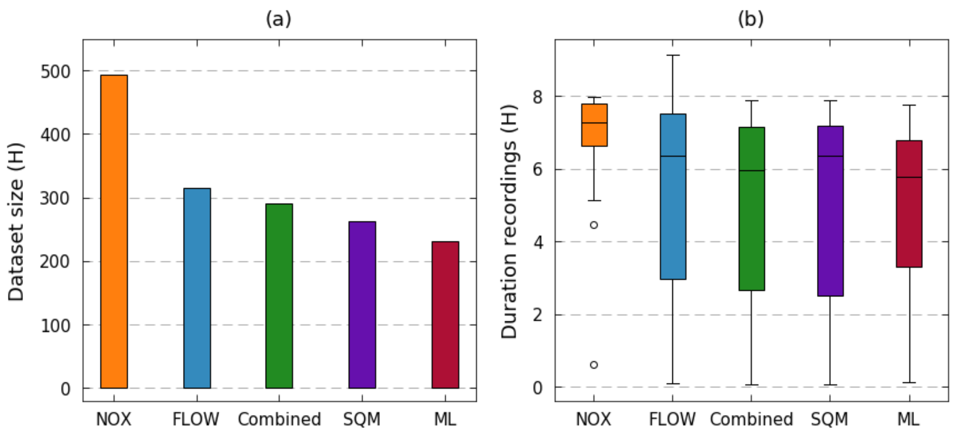

Figure 5.2: (a) Overview of dataset and (b) boxplot with the distribution of recording duration

We have successfully preprocessed and synchronized every FLOW recording except for one due to poor quality. We cannot combine every element of a recording since patients do not always start and stop recordings at the same time. As a result, there are approximately 290 hours of corresponding sleep data between the two signals that we can use for evaluation. The size of the dataset used for the signal quality metric (listed as SQM in Figure 5.2) and machine learning (listed as ML in Figure 5.2) evaluation is 263 and 232 hours, respectively. The reason the data size of SQM and ML are lower than what we have combined from FLOW and NOX is because we consider six FLOW recordings to be corrupt. Meaning these FLOW recordings are of such poor quality that even though we have been able to synchronize them, the majority of the signal is too noisy for quality measurement. Additionally, not every recording has yet been analyzed and scored by a sleep expert, which reduces the number of recordings we can use for our ML evaluation.

In Figure 5.2b, a box-plot with the duration of recordings is shown. Notice the significant difference in recording duration between FLOW and NOX. As mentioned, the FLOW sensor does not record the entire night in some cases. We know that many recordings suddenly stop in the middle of the night, which we believe is not only a sensor issue because the application used to collect the signal data from FLOW tends to crash. For comparison, NOX generally records the entire night. There are two noticeable outliers for NOX where it did not record for an entire night, represented as circles in Figure 5.2b. We believe that the patient stopped the recording themselves in these two cases, although we do not know for sure. Since many FLOW recordings do not last for the entire night, it reduces the duration of the combined and synchronized recordings, and also for evaluation.

5.2 Data Preprocessing Evaluation

The main goal of our preprocessing is to adjust the timestamp of samples in the signal from FLOW to enable synchronization with NOX. To determine the success of our preprocessing, we can visually inspect the synchronized signals from the output of our preprocessing script. After inspecting a significant number of recordings, we have yet to find any notable problems with our timestamp adjustment of FLOW samples. It is, however, a very time-consuming and tedious process to visually verify that hundreds of hours of signal data are synchronized at all-times. As such, we rely on the signal quality metric results, which we present in Section 5.3, for verifying our timestamp adjustment. Before that, we evaluate our connection loss detection.

One measurable effect of our preprocessing is that we have identified additional data as missing due to connection loss in the signal from FLOW. We have detected connection loss in 45 of the 56 recordings synchronized with NOX. Interestingly, only six recordings have recorded a complete night without having connection issues or other problems, which goes to show that it is a common problem for the majority of FLOW recordings. Overall we have detected more than 700 gaps, in the range of a few seconds long to over eighty minutes. It is important to note that not all gaps are necessarily a connection loss. For instance, in one case, the signal from FLOW keeps sending two samples every second or so, which is significantly lower than 10 samples a second at a 10 Hz sampling rate. In this case, our implementation of the window model means it continuously detects connection losses as long as new data arrive when the sampling rate is lower than expected. Additionally, the window model detects many gaps that are only a few seconds long. In many cases, the reason is that the window model is unable to adjust the timestamps of a few packets before a large connection loss.

Overall we have detected around 20 hours of missing data due to intermediate connection losses. Also worth noting is that 450 gaps have a duration of five seconds or more, and that 100 gaps are longer than one minute. This means that most connection losses are between five seconds and one minute long.

5.3 Signal Quality Metrics

Measuring the signal quality of FLOW serves multiple purposes. First of all, it confirms whether we have successfully adjusted the FLOW timestamps during preprocessing or not. Secondly, it allows us to compare the quality of overnight recordings with the signal captures in a controlled environment in [Løberg 2018]. Thirdly, the metrics may identify additional signal issues and eliminate the need to visually inspect every minute of a recording to confirm synchronization. At last, the metrics should prove useful to determine if any recordings should be considered as corrupt based on poor signal quality.

During our analysis, we discovered that it was unfeasible to measure the signal quality of hours-long recordings. The main issue is the variation in the mean baseline breath amplitude, usually related to different sleeping positions. The same issue is described in [Løberg 2018], who suggest to either compare the periods with different baseline breath amplitudes with the gold standard separately or to apply careful adaptive normalization techniques to the FLOW signal. We want to avoid applying any further preprocessing than necessary because we want to compare the quality of the raw signal to the extent it is possible. Comparing each period of different baseline breath amplitude separately is a very tedious process given the size of the dataset. As such, we have chosen to measure the signal quality periodically overnight.

We have chosen to measure the signal quality in periods of fifteen minutes. A period of this length is long enough to estimate the average breath amplitude. This length should limit the number of periods affected by motion artifacts and baseline breath amplitude changes. Measuring the signal quality periodically has a few advantages. It allows us to inspect if there is any trend in the signal quality improving or deteriorating overnight. Changes in the measured signal quality should help uncover periods with signal issues such as poor synchronization or noise, detected by changes in the different quality metrics.

5.3.1 Corrupt Signal

Early during the quality analysis, we discovered that the signal quality of FLOW sometime changes significantly during recording, likely due to belt misplacement or sensor entrapment. Most recordings contain at least one period where the signal from FLOW is very noisy. In six recordings, the FLOW signal is so noisy for several hours that we consider these recordings as corrupt. An example of a corrupt FLOW signal is shown in Figure 5.3. In this example, it is nearly impossible to visually distinguish the breaths in the FLOW signal without using the NOX signal as a reference. Therefore, we define a recording as corrupt when the majority of the signal is as poor as the example shown in Figure 5.3. This is, however, our subjective assessment of the signal that is not necessarily reflected by the quality metrics. As such, we have chosen to include the corrupt recordings in the quality measurement to see if there are any similarities. This should give us a first impression of whether the signal quality metrics can be used to determine what FLOW recordings are too poor for automatic detection of sleep apnea using ML.

5.3.2 Disrupted Breathing Events

Before measuring the signal quality of FLOW, we first present a few examples of disrupted breathing to see how the signal from FLOW behaves compared to NOX. It is, after all, fundamental that the signal from FLOW can detect breathing stops to be useful. An example of a FLOW signal of

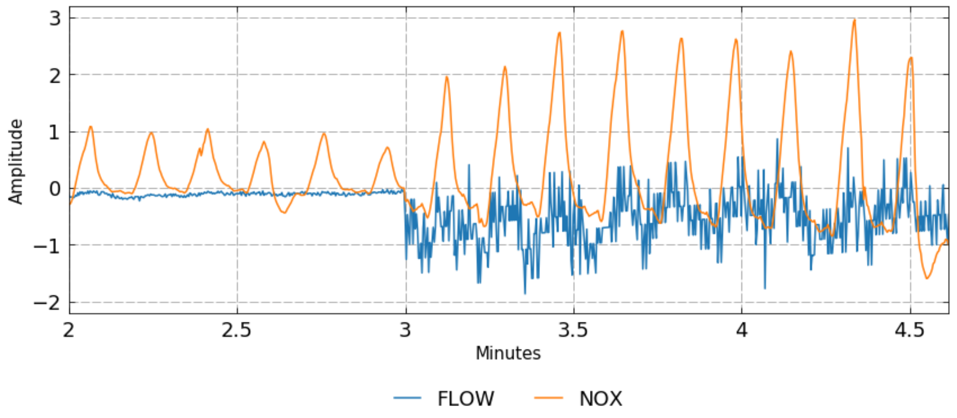

Figure 5.3: Example of a corrupt FLOW recording

very good quality can be seen in Figure 5.4 along with the NOX signal. There are five apnea events and one hypopnea event scored during these six minutes. Notice that the waveform of FLOW is almost identical to NOX during the periods of disrupted breathing. This is very promising because it means the FLOW sensor can detect the changes in breathing amplitudes similar to NOX. The apnea event around the three-minute mark seems to be a central apnea as there is a lack of respiratory effort during this period in both signals. The remaining apneas all show a presence of respiratory effort (i.e., breathing peaks), which means they are obstructive apneas.

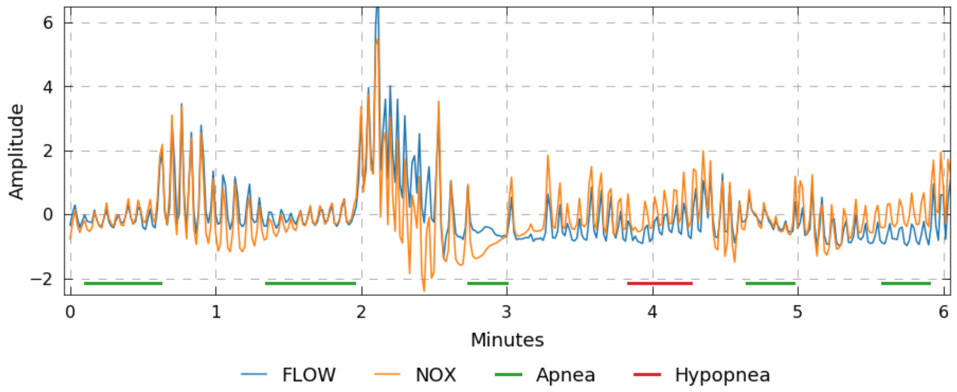

Figure 5.4: Example of a very good quality FLOW recording

Another example of the signal from FLOW during disrupted breathing is shown in Figure 5.5. In this example, the breath amplitudes in the signal from FLOW is much lower compared to the breaths in the signal from NOX. This is especially the case for breaths of lower amplitude, that is, when the patient's breathing is shallow. The FLOW signal almost flatlines which means that ML classifiers may falsely classify these periods as disrupted breathing.

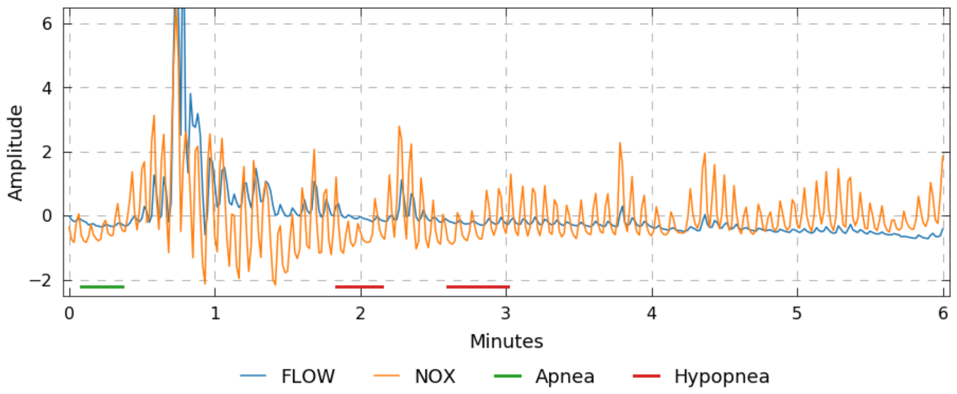

Figure 5.5: Example of a period with low mean baseline breath amplitude

5.3.3 Signal Quality Overnight

To analyze if there is any trend in the measured breath amplitude accuracy throughout the night, we have included the sensitivity and PPV in addition to the WAPE score of four recordings shown in Figure 5.6. Each *circle*, *star* or *square* in the figure represents the quality metric scores measured for a signal period of fifteen minutes. It is noticeable how much the WAPE score changes overnight in contrast to the more stable sensitivity and PPV scores. For instance, the sensitivity and PPV score of the recording in Figure 5.6a are for the majority of the night close to 100%, while the WAPE score goes from 5% and almost up to 50 % between the first and second-hour mark. At times, the WAPE score is more or less stable around 10% between subsequent periods as seen at the four-hour mark in both Figure 5.6b, Figure 5.6c and Figure 5.6d. The stability of the WAPE score in these periods, may indicate that the patient is sleeping restful and not changing their sleeping position. Notice also the low sensitivity, PPV, and WAPE score in Figure 5.6d around the seven-hour mark. In this particular period, the scores of the three metrics indicate that this signal period of FLOW is very noisy and of low quality.

The WAPE metric appears to be very sensitive in general, as the measured score often changes significantly even when the corresponding sensitivity and PPV scores indicates that the signal quality is good. The WAPE score is severely affected by motion artifacts and changes in the mean baseline breath amplitude. As a result, we cannot measure the breath amplitude accuracy of overnight FLOW recordings accurately without excluding motion artifacts and measuring each period of different baseline breath amplitude separately.

5.3.4 Breath Detection Accuracy

Sensitivity

One of the metrics we use to measure breath detection accuracy is sensitivity, which describes the proportion of real breaths correctly identified.

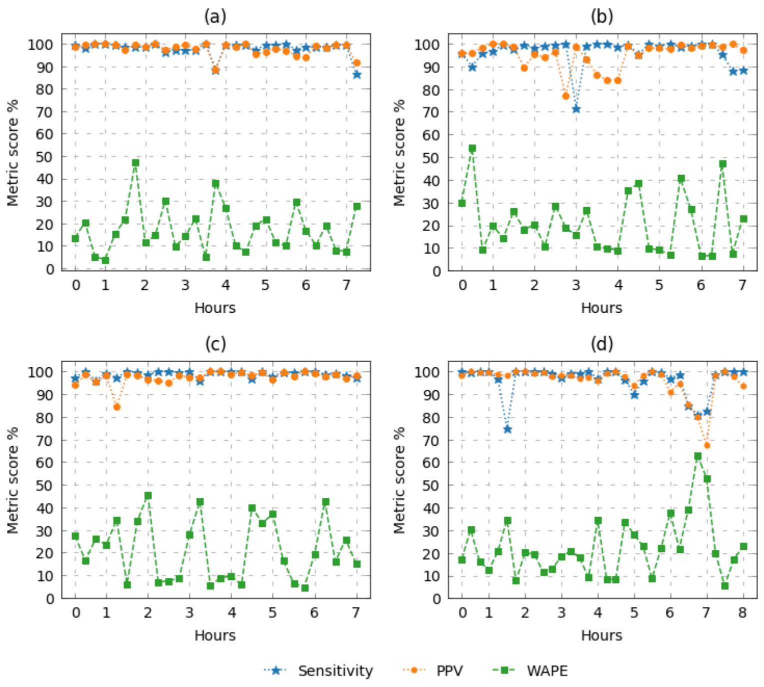

Figure 5.6: Periodically measured metric

Achieving a sensitivity score of 100% suggests that all of the real breaths are correctly identified. However, it is important to note that the possibility of false breaths mistakenly being identified as real increases in correlation to the number of false breaths. As such, sensitivity alone is not enough to imply that the signal quality is good.

The sensitivity of the FLOW recordings is very good, with a mean score of 97.2% for the non-corrupt recordings and 89.5% for the corrupt recordings. Figure 5.7 shows several box-plots with one box-plot for each recording. Each box-plot describes the distribution of the measured sensitivity scores for all periods in a recording. The median score of all the measured periods is represented with an orange line in the box-plots. The fifty-one recordings we use to evaluate the signal quality of FLOW are listed as 1 to 51, while the corrupt recordings are listed as C1 to C6.

Notice that, the sensitivity is close to 100%, during most periods for many of the non-corrupt recordings. There are a number of recordings that have a median sensitivity score similar to the corrupt recordings. See for example, recording 30 and 44 in Figure 5.7. The sensitivity of the corrupt recordings is surprisingly good, although they generally show a larger spread in the measured sensitivity compared to the non-corrupt recordings. We expect the sensitivity score to decrease as the signal quality of FLOW deteriorates past the point that breaths are no longer recognizable, since fewer real breaths should be detected. However, this is often not the case. Instead, the number of breaths detected in the FLOW signal increases, and so does the odds of

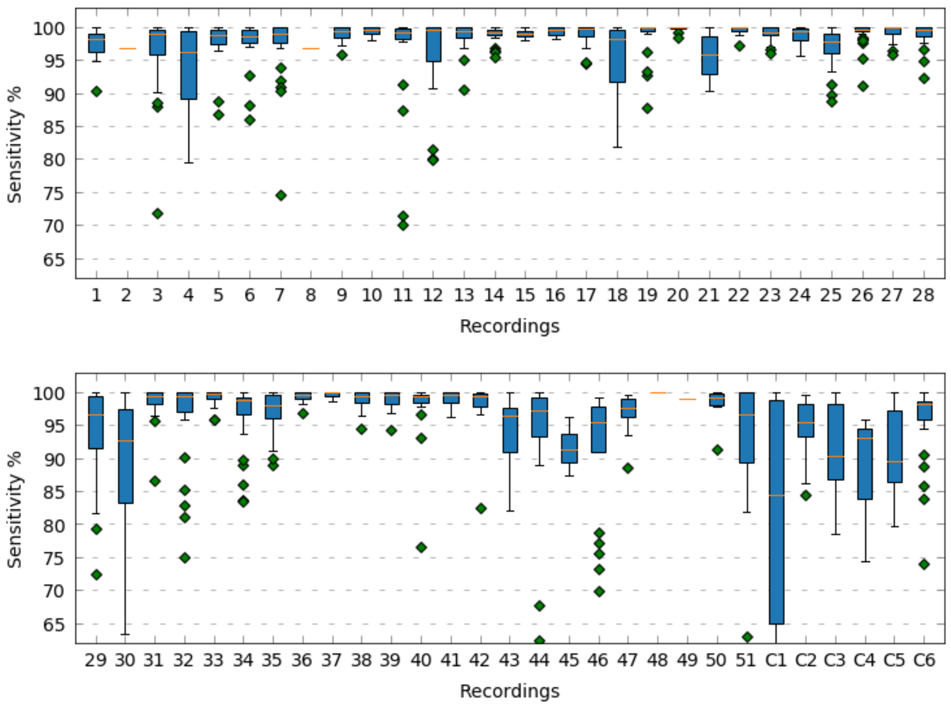

Figure 5.7: Measured sensitivity score of all FLOW recordings

false breaths mistakenly being identified as real. As a result, noise in the FLOW signal does not necessarily reduce the sensitivity score.

The sensitivity box-plots of most recordings have one or more outliers, which is a period with a significantly lower sensitivity score compared to the general distribution. In some of the recordings, FLOW appears to struggle with correctly detecting shallow breathing, although it still accurately detect deeper breaths. An example of this can be seen in Figure 5.8, where deeper breaths, that is, the breaths with high amplitudes are similar to the amplitude of breaths in the NOX signal. The shallow breaths in the FLOW signal is buried in noise and cannot easily be distinguished. This is likely because the FLOW belt is either misplaced, slightly trapped, or not fitted tightly enough on the body to register small changes in the belt expansion during shallow breathing as opposed to the larger belt expansion during deeper breaths.

Positive Predictive Value

The second metric we use for measuring breath detection accuracy is the PPV metric, which describes the percentage of detected breaths that are actual *real breaths*. Achieving a PPV of 100% means that all of the detected breaths in the are real breaths, while a PPV of 80% means that 20% of the detected breaths are false breaths. It is important to note that PPV alone is not enough to imply that the signal quality is good. For instance, it is possible to get a perfect score even if the signal flat-lines for a longer period,

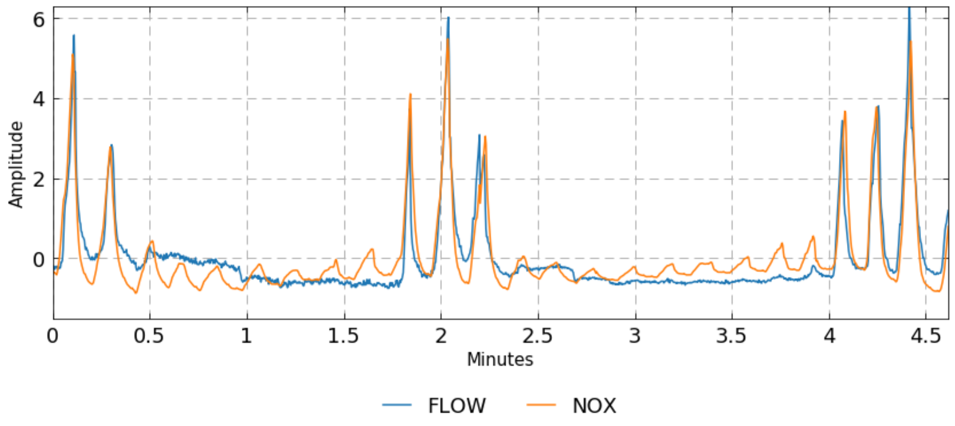

Figure 5.8: Example of FLOW not correctly registering shallow breaths

as long as the breaths detected in the remaining part of the signal are real breaths.

For most recordings, the PPV of the FLOW sensor is also very good, with the mean scores for the non-corrupt and corrupt recordings being 94.2% and 82.1%, respectively. Several of the non-corrupt recordings have a larger variation in the measured PPV score, which is more similar to the PPV scores of the corrupt recordings. See, for example, Recording 4 and 44 in Figure 5.9. The median PPV score of Recording 4 is around 80%, while four out of six corrupt recordings have a median PPV score above 85%. We know from visual inspecting Recording 4 that the FLOW signal is noisier in comparison to other recordings, but the breaths can be distinguished in the signal. Although the PPV metric does not perfectly correlate to our definition of when a recording is considered corrupt, there is more variance in the measured PPV score during the corrupt recordings in general.

Clean Minute Proportion

The third and last metric we use for measuring the breath detecting accuracy is the CMP metric, which describes the proportion of minutes in the recording where both the sensitivity and PPV are 100%. The CMP is useful for understanding the distribution of errors in the FLOW signal. If, for instance, a signal has a low PPV, the CMP can explain whether false breaths are contained within a few periods or somewhat evenly spread throughout the recording. As we measure the signal quality in periods of fifteen minutes, each clean minute increases the CMP score of that signal period with 6.66%.

The CMP score of most recordings has the largest variation. As seen in Figure 5.10, it is not unusual for FLOW recordings to have signal periods with almost no errors, that is, with a CMP score of close to 100%, in addition to a signal period with a CMP score close to 0%. The median CMP score of the corrupt recordings are centered around 10% and are among the recordings that have the lowest CMP score. The correlation between the PPV and

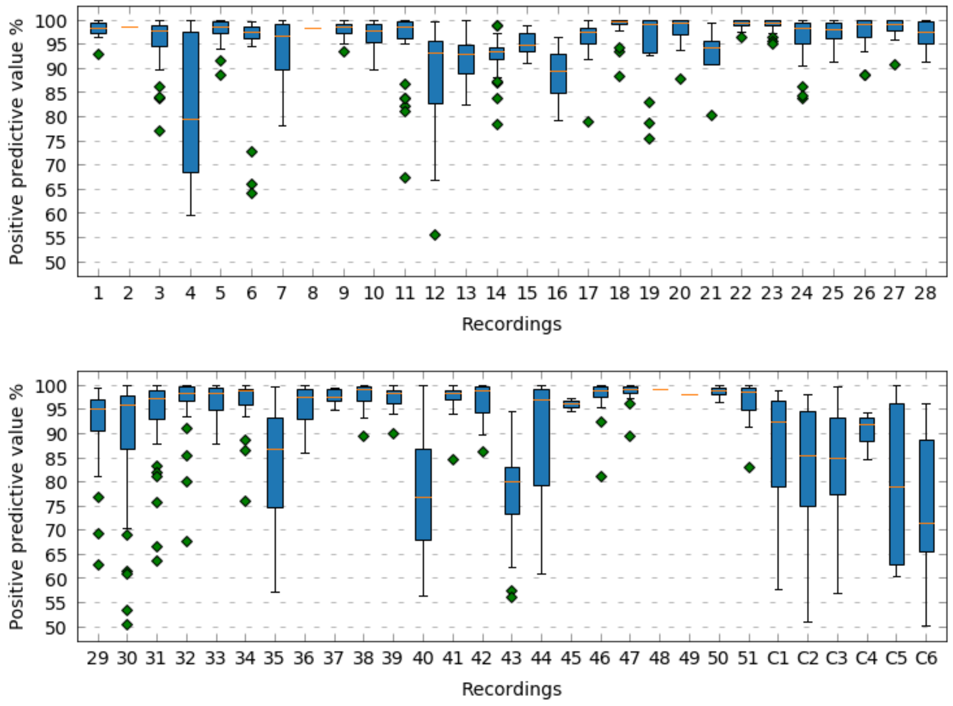

Figure 5.9: Measured positive predictive value (PPV) score of all FLOW recordings

CMP values is very noticeable and recordings with a low PPV, also have a low CMP score. See, for example, Recording 4, 12, 35, and 44, between Figure 5.9 and 5.10. The low CMP score indicates that these recordings are in general nosier for the entire night. The correlation between sensitivity is less noticeable but is still present. See, for example, Recording 18, 30, and 51 between Figure 5.7 and 5.10.

One may notice that Recording 22 and 23 have a very high CMP score. These two recordings are from the same patient, and both recordings have a sampling rate above 12 Hz. After a visual inspection of the FLOW signal, it is clear that the signal-to-noise ratio remains high both nights, possibly because the belt has been fitted perfectly tight. Notice also that Recording 4 again has an equally bad CMP score as the corrupt recordings.

5.3.5 Breath Amplitude Accuracy

We use the WAPE metric to measure the breath amplitude accuracy. This metric calculates the distance (or difference) in amplitude between corresponding breaths in FLOW and NOX. As such, a low metric score signifies a high breath amplitude accuracy, or in other words, *lower* is *better*. It is important to note that entirely random data will result in a score of 50% for this metric, which means that results close to or worse than this indicates an inferior relationship, which is one reason we are measuring the quality periodically.

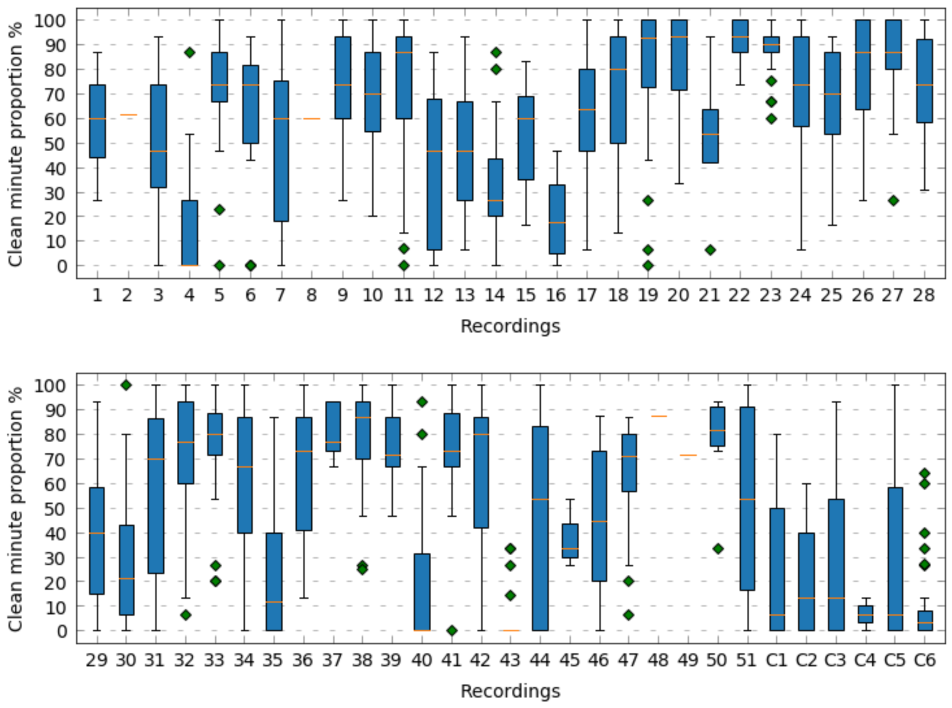

Figure 5.10: The measured clean minute proportion (CMP) of all FLOW recordings

The overall measured breath amplitude error is not great, with a mean WAPE score of 18.4%. This indicates that FLOW can only accurately detect hypopnea events with a minimum reduction in airflow of 48.4%, which is significantly higher than 30% that is the definition of a hypopnea event. Figure 5.11 shows the distribution of the periodically measured WAPE for all FLOW recordings. Notice that we cannot easily distinguish the corrupt recordings from the non-corrupt recordings using their WAPE scores compared to when using the breath detection accuracy metrics. Notice also the difference in the degree of dispersion between recordings. In fact, for most recordings, there is a noticeable correlation between the median score and the degree of variation. Generally, recordings with a high median WAPE score also have large variations, while recordings with a low median WAPE score have a more stable WAPE score.

Only eight recordings have a median WAPE score below 10%, but most recordings have measured a WAPE score below 10% in one or more periods. Recording 31 and 44 in Figure 5.11, are two examples of FLOW recordings that have one period with a very good WAPE score of 2.4% and 2.6% respectively, but also have periods with a measured WAPE score above 50%. Motion artifacts and periods with more than one mean baseline breath amplitude causes the calculated WAPE metric to be inaccurate. As a result, to determine if a period with a poor WAPE score is not because of inaccurate measurement, we have to inspect the signals.

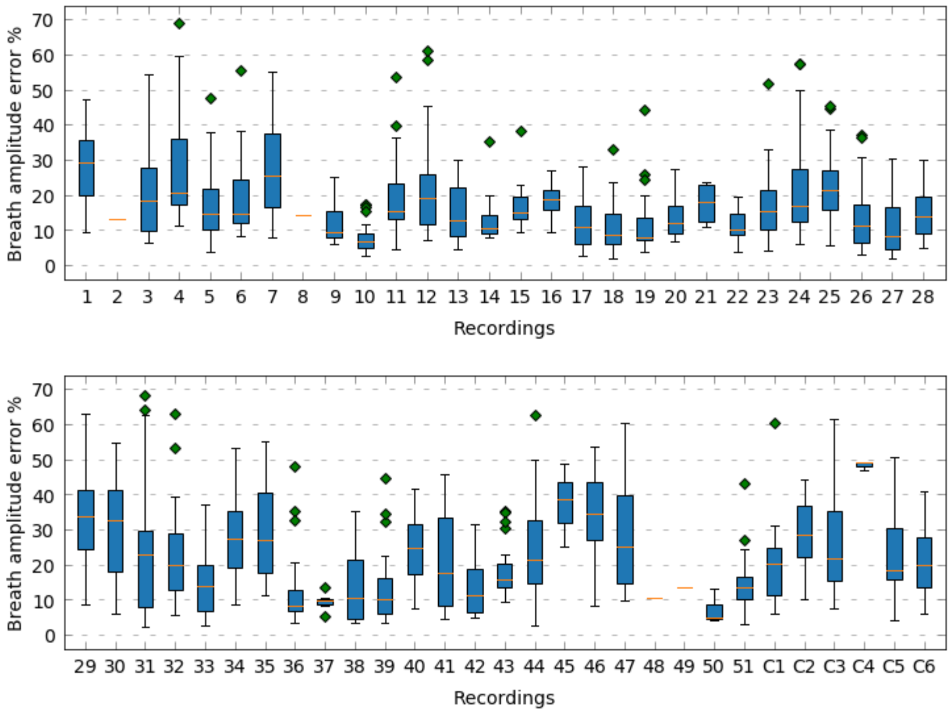

Figure 5.11: Breath amplitude error (WAPE) score of all FLOW recordings

Figure 5.12 shows the breath amplitude relationship and the regression line for a few selected periods from different recordings. We have selected these periods to show how different kinds of breath amplitude relationship between FLOW and NOX affects the quality. The periods in the scatterplot are sorted based on quality, with Figure 5.12a, Figure 5.12b, Figure 5.12c, and 5.12d having a WAPE score of 2.6%, 4.5%, 24.5%, and 38.3%, respectively.

There is not much difference between Figure 5.12a and Figure5.12b. Both periods show a strong linear relationship for shallow, normal and deep breaths between FLOW and NOX. Except for two outliers, all breaths are aligned very close to the regression line, which is what causes these periods to score so well. Together they show how similar the FLOW sensor is to NOX when it is fitted perfectly around the stomach. In Figure 5.12c, the two distinct clusters represent two different breath amplitude relationships. In this case, the regression line is fitted closest to the lower cluster, which means the breaths in the upper cluster are far from the regression line, which is the reason for the poor WAPE score of 24.5%. This illustrates the importance of measuring different breath amplitude relationships separately to determine the WAPE score accurately. By measuring the signal quality of FLOW in periods of fifteen minutes, we reduce the number of signal periods that have multiple breath amplitude relationships. However, there are still many periods with poor WAPE scores, like the one in Figure 5.12c, that should be excluded if we want to determine the actual breath amplitude accuracy

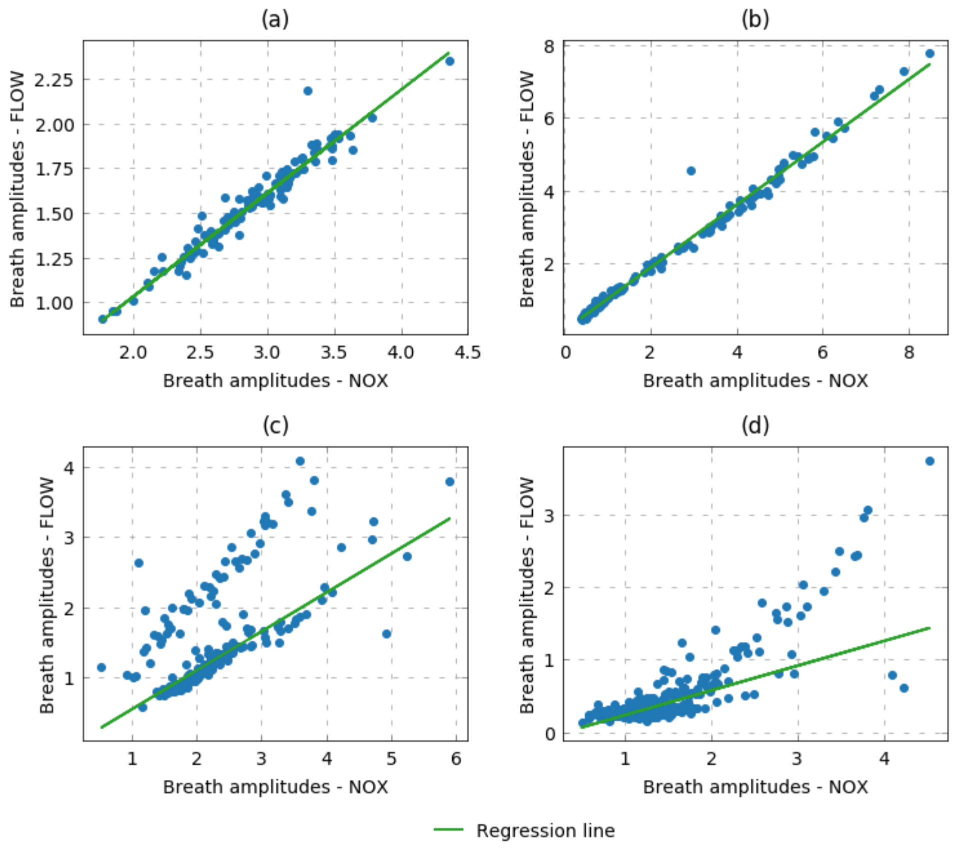

Figure 5.12: Breath amplitude relationship between FLOW and NOX

of FLOW. Figure 5.12d shows a more monotonic relationship. If we chose to exclude the shallow breaths, the slope of the regression line would be more steep, revealing a more linear relationship between FLOW and NOX for deeper breaths. This signal period is noisy, especially when the patient is breathing shallow, which indicates that the FLOW belt may not be strapped tightly enough to measure the amplitude of shallow breaths with the same linear relationship as deeper breaths.

5.3.6 Comparison with Related Work

Løberg [Løberg 2018], evaluates multiple sensors using the sensitivity, PPV, CMP, and WAPE metrics. FLOW is one of four sensors evaluated, and the RIP signal from NOX is used as the gold standard sensor. Their results for FLOW show a very good sensitivity of 98.91% for the supine position and 98.22% for the side position. Likewise, the PPV score of FLOW is very good with very few false breaths detected, resulting in a mean PPV of the supine and side positions of 98.81%, and 99.16%, respectively. FLOW is the best sensor regarding the PPV score in the study. In their study, they also find that the signal from FLOW is one of the more noisy signals among the sensors compared in the study, but that the noise is easily distinguished from breaths in general. FLOW also achieves the best breath amplitude accuracy among the sensors, with a mean WAPE score for the supine and side positions of 8.75% and 9.61%, respectively. They note that one possible explanation as to why FLOW achieves a much better breath amplitude accuracy than the other sensors is that it is the only sensor that captures the same unit of measurement as NOX (volume), meaning the signals produced by the sensors are very similar.

The performance of FLOW in our study is in line with the results of this study. Regarding sensitivity, their findings from the supine position are very similar to our results (98.81% versus 97.2%). Less similar is their results for PPV compared to ours (98.81% versus 94.2%), but our result includes a large difference in the measured PPV between recordings, ranging from 75.9% to 99.2%. There are several possible reasons as to why we experience more false breaths. First of all, we do not exclude motion artifacts. Secondly, we cannot check that the FLOW belt is attached properly, since we are not conducting the recordings in a controlled environment. Furthermore, the signal quality produced by the FLOW sensor is related to sleeping positions. If the signal quality produced is worst when sleeping on the stomach, then our results are likely to be worse, as the patients and the results are not limited to the side and supine sleeping position.

The findings in [Løberg 2018] regarding the breath amplitude accuracy of FLOW is the least similar compared to ours among the metrics used to measure the signal quality (9.61% versus 18.4%). This is somewhat expected, given our circumstances and the sensitivity of this metric. The large size of our dataset inhibits us from manually measuring periods with different baseline breath amplitudes separately, as it is too time consuming and resource demanding. However, we did find that most of FLOW recordings have a least one period with a WAPE score below 10%. Our findings indicate that the signal quality we measure periodically in overnight recordings is comparable to the results from a supervised setting.

5.4 Machine Learning

Most of the NOX recordings in this dataset have been scored by a sleep expert. As the final evaluation, this allows us to get a first impression of whether the FLOW sensor can detect sleep apnea and how NOX performs in comparison. Another benefit of machine learning (ML) is that if the classification accuracy using the FLOW signal is similar to using the NOX signal, it further confirms that our timestamp adjustment is accurate enough for the signal from FLOW to be useful.

5.4.1 Approach

Because of our limited time frame, we will evaluate eight different machine learning classifiers which are available for us to run through the CESAR project based on forthcoming work by Kristiansen et al. [Kristiansen et al. 2020], which use the same classifiers on the complete NOX T3 dataset from the A3 study. The classifiers primarily consist of deep learning and neural networks to limit the number of results and are chosen somewhat based on performance and computational efficiency.

5.4.2 Preprocessing

The annotated NOX recordings contain all of the signal data from NOX T3 in addition to the manual scorings from a sleep expert. All of this can be exported from Noxturnal to CSV files with a sampling rate of 1 Hz. We need to combine the annotated scorings with the FLOW signal. If we want to keep the sampling rate of 10 Hz, we need to upsample the annotated scorings from 1 Hz to 10 Hz. However, we encountered a problem that the NOX signal we have preprocessed and combined with FLOW deviates from the same NOX signal in the annotated file, which we cannot explain by the lower sampling rate in the annotated file alone. As a consequence, we have decided to preprocess and combine FLOW with NOX anew, using the 1 Hz NOX signal from the annotated file instead. A sampling rate reduction from 10 Hz to 1 Hz reduces the level of detail in the respiratory changes, but since nightly respiratory changes are not so sudden and typically develop over a time span of seconds, it should not have a significant impact on the results.

The annotated files distinguish between apnea and hypopnea events for both obstructive, central and mixed sleep apnea. For simplicity and since we are not trying to distinguish between different types of sleep apnea, we have decided to combine all events related to either hypopnea or apnea into one class for general disrupted breathing, as seen in Table 5.1. Each row of this class is denoted as either 1 for disrupted breathing, or 0 for normal breathing.

	Timestamp	FLOW	NOX	Class
1	2018-09-04 23:57:08	0.147436	0.218308	0
2	2018-09-04 23:57:09	-0.518685	-0.784432	0
3	2018-09-04 23:57:10	-0.539831	-1.141768	0
4	2018-09-04 23:57:11	0.496357	-0.395067	1
5	2018-09-04 23:57:12	0.972158	1.075836	1

Table 5.1: Example of annotated file

We have chosen to apply another preprocessing step to the signal from FLOW to adjust for baseline shifts and periods with low mean breath amplitude. Currently, this is the most obvious difference between the FLOW and NOX signals, although other types of preprocessing, such as noise removal, also may increase the signal quality. One reason for doing this is that we compare the ML results of different levels of preprocessing, which will give us some indication about the impact further preprocessing can have on classification performance regarding data from FLOW.

To remove motion artifacts and correct the baseline shifts, one option is to use the ABAMAR algorithm presented in Section 3.3.4. However, this algorithm does not consider the impact that periods of different breath amplitudes possibly have on the performance of ML classifiers. Moreover, it needs to be carefully adapted to reach the FLOW signal, which is time consuming. Instead, we have chosen a simple approach to standardize the FLOW signal in small periods of one minute. As shown in Figure 5.13, the difference from standardizing the signal every six-minute, see Figure 5.13a, compared to every minute, see Figure 5.13b, is clear. The FLOW signal in Figure 5.13a almost flatline towards the end, which can confuse a data mining classifier to mistakenly detect this period as disrupted breathing. However, when we standardize the signal every minute, it is obvious that the signal is not flatlining and now looks more similar to NOX.

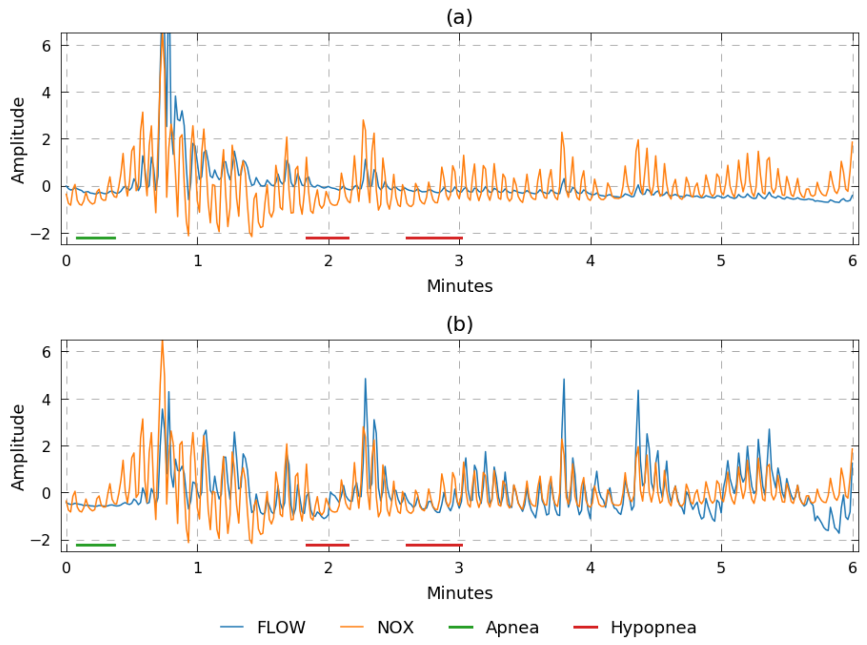

Figure 5.13: Amplitude difference from standardizing every sixth minute versus every minute

5.4.3 Results

The performance of classifiers is often related to the size of the dataset used. Therefore, we test each classifier using different sizes of the dataset. To differentiate between the results of a classifier, we append the size of the dataset to the name of the classifier. We use three sizes of the dataset, defined as S for small, M for medium, and L for large. For instance, the results of running the convoluted neural network (CNN) classifier using the large dataset is represented in the following figures as CNNL.

We start in Figure 5.14, by comparing the kappa scores to get a better insight into the general performance of the different classifiers, as shown in Figure 5.14. A kappa value of 1 indicates a perfect agreement between the underlying metrics. The circle/point in the middle of each line is the mean kappa score of the 10-folds, while the line represents one standard deviation (SD). The SD indicates whether the kappa of each fold is close to the mean (low SD) or more spread out over a wider range (high SD). Notice that most of the classifiers have a much lower kappa score on the FLOW signal without additional preprocessing. For instance, all sizes of random forest (RF) have a kappa value close to 0, which means that there is a lack of agreement between the metrics used to evaluate the classifier. The only two feedforward neural networks we evaluate, CNN and MLP, are capable of feature extraction on their own and manage to achieve a kappa score only slightly lower than with our preprocessing. For most classifiers, the signal from NOX generally has a kappa score between 0.55 and 0.6, while the preprocessed signal from FLOW is centered just below 0.5. This suggests, that the classifiers perform slightly better on signal data from NOX, than from FLOW.

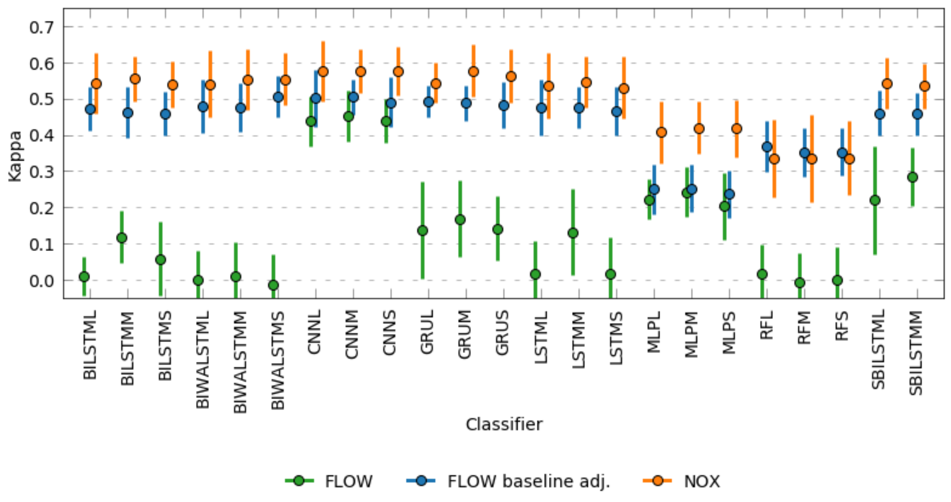

Figure 5.14: Average and SD results of kappa

Figure 5.15 shows the results of the accuracy metric of all classifiers. For most of the classifiers, the accuracy reflects the kappa results to a large degree. The accuracy of the recurrent networks, like the long short term memory (LSTM), on the signal from FLOW without further preprocessing, is generally centered around 50%, and since the dataset is balanced with only one class, the results are no better than random guessing. The medium-sized CNN (see CNNM in Figure 5.15), has the highest accuracy for both FLOW, the further preprocessed FLOW signal, and the NOX signal, with a mean accuracy of 73.3%, 76.1%, and 79.6%, respectively. We are able to improve the accuracy of FLOW with a simple standardization even with an initial accuracy as high as 73.3% using CNN. This indicates that there is a strong correlation between the features we focus on, namely the amplitude, and the scoring of the sleep expert.

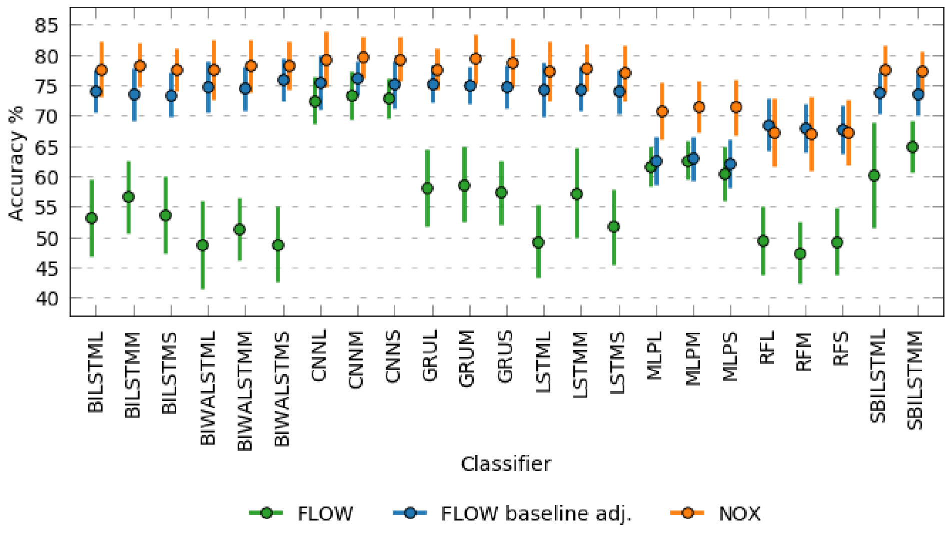

Figure 5.15: Average and SD results of accuracy

The measured sensitivity of most of the classifiers has the highest spread between the different folds. This is worst for the FLOW signal without further preprocessing, as seen in Figure 5.16. The sensitivity of NOX is around 3-4% better than the preprocessed FLOW signal. CNNL has the best results for FLOW, preprocessed FLOW, and NOX with a mean of 71.5%, 81.1%, and 84.3%, respectively. A high sensitivity is important as it represents the classifiers' ability to detect disrupted breathing. However, a classifier can get a perfect sensitivity simply by detecting everything as disrupted breathing. Therefore, this metric on its own is not enough to evaluate the performance of a classifier. So far, CNN has scored overall the best for three metrics indicating that the classifier is very suitable for the detection of disrupted breathing.

The specificity of the classifiers is shown in Figure 5.17. Notice that the signal from FLOW in some instances has a higher score compared to the preprocessed FLOW signal. See, for example, CNN and MLP in Figure 5.17. Of the metrics we present, this is the only one that is somewhat negatively affected by our further preprocessing of the FLOW signal. The classifiers that experience a decrease in specificity score after further preprocessing of FLOW are the same classifiers that experience an increase in their sensitivity score. Our standardization of the FLOW signal should ideally increase the breath amplitude in periods with a lower mean breath amplitude. As a side effect, this procedure can reduce the breath amplitude in other signal parts, such that they are mistakenly identified as disrupted breathing, which reduces the specificity score. There is a correlation between the measured sensitivity and specificity, with classifiers having a lower sensitivity have a higher specificity, see for example MLP between Figure 5.16 and Figure 5.17.

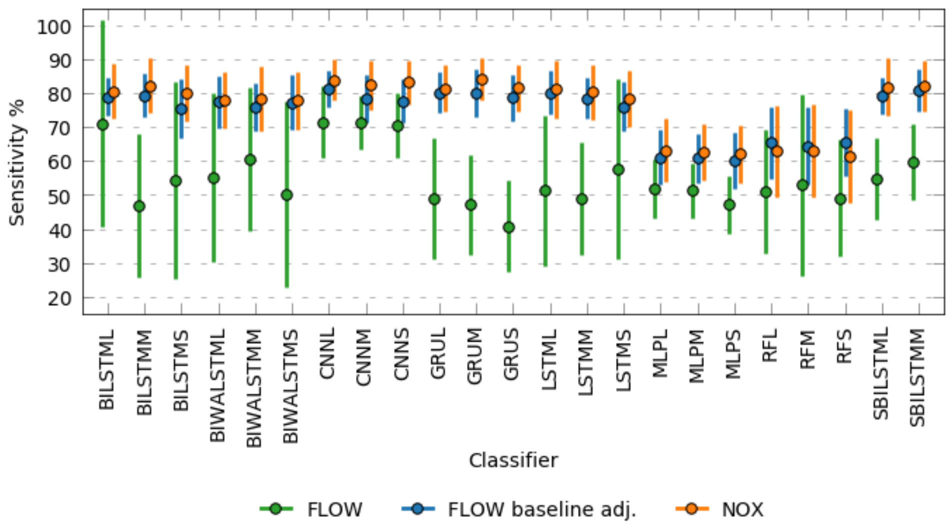

Figure 5.16: Average and SD results of sensitivity

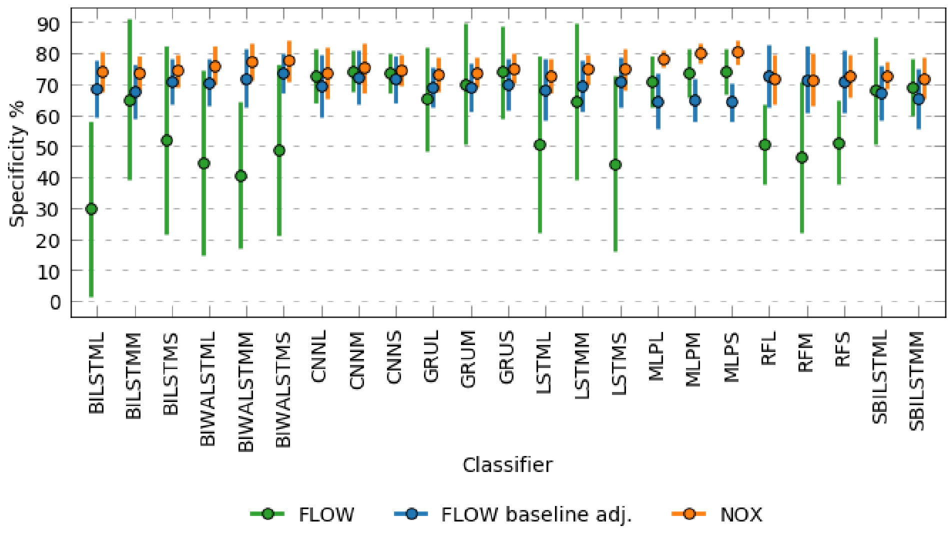

Figure 5.17: Average and SD results of specificity

5.4.4 Comparison with Related Work

A direct comparison with related work on the classification of polygraphy (PG) sleep data using consumer grade sensors is not available, to the best of our knowledge. However, we can compare the results from the forthcoming paper [Kristiansen et al. 2020], which uses the same set of classifiers as us. In their study, they use the complete NOX T3 dataset from the A3 study, which means they are not limited to the abdomen signal, but also include signals such as the respiratory effort from the chest (thorax) and cannula airflow, among others. In [Kristiansen et al. 2020], CNN overall achieves an accuracy of 83.4%, a sensitivity of 82.6%, and a specificity of 84.2% using the abdominal signal from NOX. The CNN results are also the average from using a small, medium, and large size of the A3 dataset.

Some of the patients in the A3 study were given the opportunity to use the FLOW sensor in addition to NOX T3 while sleeping. Our findings for NOX should be similar since our dataset is a subset of the A3 study of patients that used both belts. However, the average accuracy of the CNN classifier of 79.6% on our data from NOX is slightly lower than that of 83.4% in [Kristiansen et al. 2020]. Since the size of the two datasets is the only major difference, this indicates that the performance of CNN is related to the size of the dataset. Considering that a larger dataset increases the accuracy of CNN with almost 4% on NOX data, it is fair to assume that the accuracy of FLOW also increases with a similar rate. Therefore, it is very likely that the accuracy of the CNN classifier will increase from the current 76.1% towards 80% with a larger dataset.

5.5 Discussion and Conclusions

Preparing FLOW Data

It is not trivial to prepare data from FLOW for evaluation, in fact, it is very challenging. Even then, there is still no guarantee that everything has worked properly with synchronization and gap detection. There are aspects that can affect the data quality, which is hard to check by any other means than visual inspection of the recordings.

FLOW Issues

During this work, we have discovered two areas of major issues with the FLOW sensors. The first area is associated with communication and application development. This includes Bluetooth connection, data loss, gap detection and wrong timestamps. The second area is to our understanding related to the very structure of the sensor itself, which *probably* is more prone and sensitive to motion artifacts, which again *probably* leads to baseline shifts. This is our assumption based on the fact that the NOX T3 sensor stretches the whole circumference of the abdomen, which means that the effect from changing sleeping position is less significant. Moreover, some amount of automatic preprocessing is likely applied to NOX T3 recordings by the Noxturnal software program. As a result of the first area, we have significantly less data than we could have had. The second area leads to a lower quality of the data we have than one could have wished for, for good classification. The reason for this is the large difference in baseline breath amplitude and a lack of extensive preprocessing to correct this. In conclusion, the differences and similarities between the two signals and their data are important findings that allow for further improvement of the quality of the FLOW sensor.

Signal Quality and Timestamp Correction

Determining the signal quality and the success of our timestamp correction cannot be seen on a recording as a whole. Instead, a window-based solution is needed such that sudden quality changes can be identified and inspected. The quality metrics are based upon NOX as our gold standard that represents the ground truth. We have, however, discovered a number of instances where this is not the case, where the NOX sensor is malfunctioning by flatlining or limiting the peaks of breath, while the FLOW sensor is working as expected. The unattended overnight recordings are likely to contain periods with motion artifacts due to restless sleeping or people waking up to use the bathroom in the middle of the night. Such periods should not be included during the quality measurement of FLOW. While a window-based solution gives us an indication of the general quality, the actual quality is likely better than what the metrics suggest, given the limited amount of data cleaning we apply. What remains important is that the signal of FLOW behaves in a distinguishable way during periods of disrupted breathing so that machine learning classifiers can learn those patterns to differentiate between periods of normal or disrupted breathing.

Synchronizing Signals

Synchronization of FLOW and NOX has proven to be relatively easy with a good timestamp correction of FLOW samples. A less precise timestamp correction would likely have increased the difficulty of maintaining synchronization overnight. For synchronizing the signals, we are simply comparing two amplitudes from different signals. As long as the signal quality of FLOW is good enough that breaths are distinguishable from noise, it is not a problem to synchronize the signal with NOX.

WAPE Metric Issues

One of the main problems with measuring the signal quality is related to the sensitivity of the WAPE metric used to measure the breath amplitude accuracy. The challenge arises as we determine the linear relationship between breath amplitudes to derive the regression line, but multiple relationships exist. As suggested in [Løberg 2018], periods of different baseline breath amplitudes should be measured separately, or else the relationship is completely misleading. However, separating different baseline breath amplitudes for overnight recordings is not feasible given the sheer volume. The window-based measurement is a compromise that limits how many windows are affected by multiple breath amplitude relationships, so the overall WAPE score of recordings is less affected. However, the problem is present, which means that the overall WAPE metric cannot precisely measure the breath amplitude accuracy of overnight FLOW recordings without manual excluding the affected periods.

For FLOW, the breath amplitude relationship changes between linear and monotonic overnight, possibly based on the sleeping position and belt placement. Since the sensor part of FLOW only spans a small part of the belt, the sensor is more likely to become trapped or misaligned, especially if it is not fitted properly in the first place. In some recordings, the FLOW signal struggles with registering shallow breaths with the same linear relationship as deeper breaths. This illustrates the need for further analysis of the impact of belt placement on measurement quality and classification performance.

Corrupt FLOW Recordings

Besides the fifty-one nights of recording we use to evaluate the FLOW sensor, we include six recordings considered as corrupt, to determine if there is any correlation between our definition of a corrupt recording and the quality metrics. The sensitivity of the corrupt signals compared to the non-corrupt signals (89.5% vs. 97.2%) alongside the PPV (82.1% vs. 94.2%) and CMP scores (21.4% vs. 59.4%), indicates that the recordings we consider corrupt, indeed are worse than average, according to the breath detecting accuracy metrics. However, if we inspect Figure 5.7, Figure 5.9 and Figure 5.10, the metrics cannot easily separate corrupt recordings from recordings which we have determined not to be corrupt. Although the WAPE metric of the corrupt recordings is also lower than the non-corrupt (24.4% vs. 18.4%), there is no clear correlation between the WAPE metric and our definition of a corrupt recording.

The recordings that we determine to be corrupt is based on our subjective assessment of the signal quality. The quality metrics should give a more objective assessment of which recordings are likely to be corrupt, although it is too soon to conclude anything specific. Therefore, it remains an open issue if the signal quality metrics can be used to decide if a recording is useful for machine learning or not. To answer this question, we need to collect a larger dataset with more corrupt or poor recordings to balance the dataset.

Machine Learning

Our initial evaluation of eight classifiers reveals the importance of further preprocessing of FLOW for automatic detection of disrupted breathing. Feedforward neural networks like CNN and MLP, are to a large degree, able to extract the important features themselves, while our primitive standardization of FLOW increases their accuracy with additional 3%. Although recurrent neural networks are similarly advanced as feedforward networks, it is interesting that their results on the FLOW signal without further preprocessing is often not better than random guessing. The exact reason remains unknown and is an issue for future work.

Further Preprocessing

Since this is the first study where FLOW sensors are used in a clinical setting on real patients, it is very promising that the CNN classifier achieves an accuracy of 73.3% on FLOW data and 76.1% after applying further preprocessing. Compared to the accuracy of 79.6% CNN achieves on NOX data, the difference in classification performance on the two sensors are only 3.5%. The classification result from most classifiers is significantly better than random guessing, and the few simple measures taken concerning baseline shifts and breath amplitudes indicate that with a comprehensive preprocessing scheme, the classification performance is likely to increase. Considering the issues with FLOW, our accuracy findings are important as it shows that low-cost sensors can achieve a similar accuracy to medical grade sensors.

Further Classification

Excluding recordings of poor quality will most likely increase classification performance to some extent, even more, if we only use the recordings with the best quality. The recordings we have obtained are comparable with what people will record at home in the future, which means that the quality of some recording is likely to be poor. Training a classifier on poor quality data limits the performance that can be achieved. Therefore, it is important to study how the quality of FLOW data affects the classification performance.

Measuring Data Quality without NOX T3

In the future, people will not be using the NOX T3 during an initial sleep apnea test at home. Therefore, we need another way to determine the quality of recordings to limit misclassification. The obvious solution is to train a classifier to identify recordings of lower quality. A potential benefit from this is that people can get feedback after a night of recording, which likely increases the quality of future recordings as they get used to wearing the sensor and learn how to use it correctly.

Traditional Scoring Rules

In traditional sleep diagnosis it is normal to operate with a minimum requirement for duration of recordings. Since we are trying to provide people with the option to perform an initial sleep apnea test at home, we are not limited by the typical standards used in the field. However, to determine which people to recommend for a further check-up, some sort of metric is required. Typically the AHI-index is used to indicate the severity of sleep apnea and is calculated based on the total sleep time. During traditional PSG there are several signals used to determine whether a subject is actually sleeping, but this will not be the case during an initial sleep monitoring at home. Therefore, it is important to consider the impact of including large proportion of data during wakefulness on classification.

Chapter 6

Conclusion

In this chapter, we conclude the work of this thesis. We begin in Section 6.1, with a summary of our main contributions. This includes the process of identifying and correcting the issues with the FLOW sensors, evaluating the signal quality and preparing the dataset for an early evaluation of eight different classifiers' performance for detecting disrupted breathing. We continue in Section 6.2, with a critical assessment of our work, before we end the thesis by suggesting a few possibilities for future work in Section 6.3.

6.1 Summary of Contributions

In this thesis, we analyze the *usefulness* of the FLOW sensor during unattended overnight sleep monitoring at home. The problem statement from Section 1.2 and their answers are provided in the following subsections:

- the quality of the collected data is answered in Section 6.1.2,
- how good and consistent the signal quality from FLOW sensors is during unattended overnight sleep monitoring at home is answered in Section 6.1.3,
- the performance of ML classifiers on the collected data and if additional preprocessing increases the performance are answered in Section 6.1.4

Before tackling the overall problem statement of whether ML classifiers can detect disrupted breathing from FLOW data, several problems are required to be solved. Many of the partial problems that are extremely difficult to solve were not predictable beforehand, which means they are not explicitly a part of the problem statement. These issues are generally related to the cleaning and preprocessing of data from FLOW.

As the first contribution in this work, we have collected the data from both FLOW and NOX recordings and conducted an initial *data cleaning*. This mainly involves identifying the real recordings and removing several tests or unsuccessful recordings. Given the uniqueness of the data and lack of metadata information, a significant amount of time was also spend locating missing recordings from both FLOW and NOX.

One of our main contributions is an early analysis of the sensor problems related to FLOW. We discovered three specific issues regarding abnormalities in the sampling rate, unreliable time-stamping, and temporary loss of connection. These issues, in combination, create a problem that proves difficult to solve. As a result, we include a discussion of the following possible alternatives for preprocessing FLOW recordings, which is not included in the problem statement:

1. Assume the sensor produces samples with a constant rate of 10 Hz based on the given information from SweetZpot.
2. Estimate the sampling rate for each recording.
3. Estimate the sampling rate periodically throughout each recording.
4. Assume the timestamp of the last sample in each packet is correct.
5. Use a jumping window model.

As a consequence of the insufficiency of the first four alternatives tested, the solution is the fifth option, which is the window model we design. The window model is able to adjust the timestamps with a good enough precision to maintain synchronization overnight while also detect connection losses. For machine learning (ML), we address the signal problems in FLOW regarding baseline shifts and changes in mean breath amplitudes. The additional preprocessing applied is a standardization of the FLOW signal on a minute window-based approach.

6.1.1 Window Model

The design of the window model is generic to accommodate for higher or lower sampling rate by changing the parameters. The current solution can handle oversampling because all samples in a window are distributed evenly in the window. It is more challenging if the sensor undersamples, which can cause the model to detect a connection loss. To avoid this, one would need a more complex model than ours that changes the expected sampling rate throughout the recording, which can be based on the function we use to estimate the overall sampling rate of a recording.

6.1.2 Quality of Dataset

The initial quality of the collected data cannot be considered useful or fit for our purpose. The steps we take to correct the major issues with FLOW has improved the quality such that the recordings can be synchronized and are useful for further analysis. As this is the first study where FLOW sensors have been used in an unattended setting, some sensor problems are likely to occur, which affects the quality of the data we collect. For instance, the synchronization would be accurate if the FLOW sensor timestamped the samples. Instead, we have to estimate the timestamping, which is not as accurate. There is no guarantee that everything has worked properly with preparing data from FLOW, which means there are still possible aspects that can affect the data quality, such as periods with poor timestamp adjustment.

6.1.3 Signal Quality of FLOW

We have made some minor adjustments to two scripts in [Løberg 2018] to accommodate synchronization and quality measurement of overnight recordings with gaps. We evaluate the signal quality of fifty-one overnight recordings using a window-based approach. The main reason is to reduce the impact that baseline shifts and different mean breath amplitudes have on the results from the more sensitive WAPE metric. However, there are still many periods where the WAPE metric is affected by motion artifacts and multiple breath amplitude relationships between FLOW and NOX. Nevertheless, the WAPE metric is still useful for determining when the FLOW sensor is registering shallow breaths with a very low amplitude as it is common that FLOW recordings have periods where breaths are barely distinguishable from noise. This implies that the signal quality of FLOW is not always consistent overnight and can suddenly change from good to bad or vice-versa depending on several factors such as sleeping position, belt placement, and restless sleep. FLOW achieves a sensitivity, PPV, CMP, and WAPE metric score of 97.2%, 94.2%, 59.4%, and 18.4%, respectively. In comparison, the six corrupt recordings achieve a lower sensitivity, PPV, CMP, and WAPE metric score of 89.5%, 82.1%, 21.4%, and 24.4%, respectively.

6.1.4 Machine Learning Classification

We evaluate the performance of eight different data mining classifiers on forty-five overnight recordings. The classifiers we use, are readily available to us through the CESAR project and are based on a fourth-coming paper [Kristiansen et al. 2020]. We compare the classifiers on the NOX and both before and after standardizing the FLOW signal. Among the eight classifiers, CNN is generally the classifier that has the best performance on both the raw and preprocessed signal from FLOW and NOX. Feedforward neural networks like CNN are able to do feature extraction to a large degree, while our preprocessing further increases their accuracy with 3%. This confirms that preprocessing indeed increases the performance of classifiers. Meanwhile, the performance of recurrent neural networks like LSTM in many cases is not better than random guessing on the not standardized FLOW signal. With a simple preprocessing scheme like ours, we are able to improve the accuracy of FLOW to a 4% difference from NOX. Between the three different data sizes of CNN, FLOW achieves an accuracy, sensitivity, and specificity of 75.6%, 78.9%, and 71.1%, respectively, while NOX achieves an accuracy, sensitivity, and specificity of 79.4%, 83.1%, and 74.5%, respectively. This suggests that low-cost sensors can achieve a similar classification accuracy to medical-grade sensors, especially when we take into consideration the sensor issues we experience with FLOW.

6.2 Critical Assessment

With more than three hundred hours of sleep data and the limited time frame in mind, there are inevitably some errors in the dataset we have not discovered. With most datasets, some amount of data cleaning and preprocessing is required. However, we spend a significant amount of time on the sensor issues with FLOW and combining the FLOW and NOX recordings in the dataset, which means that less time was available for further preprocessing to improve the signal quality and for further evaluation of the performance of classifiers on FLOW.

To limit the complexity of the window model, we have chosen not to consider variations in the sampling rate throughout a recording. The main reason being that the results of the current solution have proven sufficient for our needs. Since some of the FLOW sensor problems discovered in this work already have been addressed by SweetZpot, it is unlikely that new FLOW recordings require a more complex solution than the current model.

The design of the window model is based on the knowledge and experience we have gained from performing a significant amount of tests of various solutions. Another person with a different background may have come up with a different approach and similar results. To the best of our knowledge, there are no algorithms that is easily adaptable to the specific set of problems we face. An important matter is that we rely on the timestamps of another sensor as the ground-truth. It is easy to forget that this signal also can have flaws like wrong time-zones and signal flatlining that we have discovered.

6.3 Future Work

Given our limited time frame, there are a number of opportunities for future work related to our dataset. It remains an open issue if the signal quality metrics used in this work strongly correlates to classification performance. If this is the case, it may very well be possible to determine whether a recording or parts of a recording, is good enough for classification based on the signal quality and when recordings should be considered corrupt.

In this work, we have barely touched upon the vast amount of options for improving the signal quality. Besides a simple window-based approach of standardizing the signal once every minute, we have not applied any additional preprocessing. Considering that this is enough to improve the accuracy of most classifiers with at least 3%, it is likely that more advanced techniques are capable of improving the quality even further and minimizing the difference between FLOW and NOX even more. One option is to use the ABAMAR algorithm suggested in [Virtanen et al. 2011] to detect motion artifacts and baseline shifts using the accelerometer already present in the

FLOW sensor and do the same. Another option for further preprocessing is to look into feature extraction, since optimizing the selection of data features can increase the performance of classifiers.

Exploring different types of ML classifiers and whether fine-tuning different parameters can increase the overall accuracy is an option for further study. Another aspect regarding classification that remains an open issue is why feedforward neural networks perform better on data that have not been preprocessed compared to recurrent neural networks.

Investigating if there is any correlation between sleeping position and the signal quality of FLOW is another open issue. As NOX T3 and Noxturnal software program contain information about sleeping positions, it may be possible to determine what is the dominant factor affecting the quality of FLOW overnight. Understanding whether a specific sleeping position causes belt entrapment or if it is more related to poor belt placement, is an important aspect for further study. One reason is that we may want to exclude parts of the recording where the signal almost flatlines to avoid misclassification during ML.

Related to this is a general analysis of the implications of belt placement for classification performance. To do this, one option is to use future versions of the FLOW sensor, which hopefully includes timestamping on the sensor and local data storage. This should remove the sensor issues we have now and allow for a more in-depth analysis of the signal quality of FLOW.

Since this is a preliminary study of the FLOW sensor, and the fact that we have promising results, the natural cause of direction is to collect more overnight sleep recordings with the FLOW sensor to increase the accuracy of ML methods. A larger dataset will allow the possibility to study the correlation between the signal quality metrics and classification performance even further. Some improvements have already been made to the smartphone application based on our findings, resulting in connection losses being more easily identified, which will allow for a smoother preprocessing in the future.

Considering that the primary long-term goal of the CESAR project is to allow people to take the first step towards a sleep apnea diagnosis at home, it is important to determine when a recording is of good enough quality to be used for classification. For instance, the premise for detecting disrupted breathing is that the person is sleeping, which we need to determine based on the recorded data. One option is to use the heart-rate monitor and accelerometer in FLOW to estimate when a person is sleeping. Since people will not be wearing the NOX T3 in the future, it is also important to find another way to determine the signal quality of FLOW. One possible solution is to train a classifier to detect low-quality recordings using FLOW recordings we know are of poor quality.

Bibliography

Alaska Sleep Clinic (2018a). "The 3 Types of Sleep Apnea Explained: Obstructive, Central, and Mixed". Accessed February 24, 2020. URL: https://www.alaskasleep.com/blog/types-of-sleep-apnea-explained-obstructive-central-mixed.

— (2018b). "The 4 Most Common Sleep Disorders: Symptoms and Prevalence". Accessed March 11, 2020. URL: https://www.alaskasleep.com/blog/the-5-most-common-sleep-disorders-symptoms.

Alpaydin, Ethem (2014). *Introduction to Machine Learning*. 3rd ed. The MIT Press. ISBN: 9780262028189.

Alvarez-Estevez, Diego (2020). "European Data Format". Accessed March 5, 2020. URL: https://www.edfplus.info/.

ASAA (2020a). "Obstructive Sleep Apnea". Accessed February 23, 2020. URL: https://www.sleepapnea.org/learn/sleep-apnea/obstructive-sleep-apnea/.

Askham, N, D Cook, M Doyle, H Fereday, M Gibson, U Landbeck, R Lee, C Maynard, G Palmer and J Schwarzenbach (2013). "The Six Primary Dimensions for Data Quality Assessment". *Group, DAMA UK Working: 16*.

Berry, Richard B., Rohit Budhiraja, Daniel J. Gottlieb, David Gozal, Conrad Iber, Vishesh K. Kapur, Carole L. Marcus et al. (2012). "Rules for scoring respiratory events in sleep: Update of the 2007 AASM manual for the scoring of sleep and associated events". In: *Journal of Clinical Sleep Medicine* 8.5, pp. 597–619. DOI: 10.5664/jcsm.2172.

Bronstein, Jason Z. and Lee J. Brooks (2017). "A potential alternative to respiratory inductance plethysmography for children?" In: *Journal of Clinical Sleep Medicine* 13.2, pp. 159–160. DOI: 10.5664/jcsm.6430.

DZone (2010). "Map Reduce and Stream Processing". Accessed March 26, 2020. URL: https://dzone.com/articles/map-reduce-and-stream.

Gottlieb, Daniel J., Jeffrey M. Ellenbogen, Matt T. Bianchi and Charles A. Czeisler (2018). "Sleep deficiency and motor vehicle crash risk in the general population: A prospective cohort study". In: *BMC Medicine* 16.1, p. 44. DOI: 10.1186/s12916-018-1025-7.

Hrubos-Strøm, Harald, Anna Randby, Silje K. Namtvedt, Håvard A. Kristiansen, Gunnar Einvik, Juratešaltyte Benth, Virend K. Somers et al. (2011). "A Norwegian population-based study on the risk and prevalence of obstructive sleep apnea The Akershus Sleep Apnea Project (ASAP)". In: *Journal of Sleep Research* 20.1 PART II, pp. 162–170. DOI: 10.1111/j.1365-2869.2010.00861.x.

Huang, Qi Rong, Zhenxing Qin, Shichao Zhang and Chin Moi Chow (2008). "Clinical patterns of obstructive sleep apnea and its comorbid conditions: A data mining approach". In: *Journal of Clinical Sleep Medicine* 4.6, pp. 543–550. ISSN: 15509389. DOI: 10.5664/jcsm.27348.

Kristiansen, Stein, Morten Andersen, Vera Goebel and Thomas Plagemann (2020). "Comparing and Analysing the Flow and NoxT3 Sensor Signals for Sleep Apnea Detection". *work-in-progress*.

Kristiansen, Stein, Mari Sønsteby Hugaas, Vera Goebel, Thomas Plagemann, Konstantinos Nikolaidis and Knut Liestøl (2018). "Data Mining for Patient Friendly Apnea Detection". In: *IEEE Access* 6, pp. 74598–74615. ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2882270.

Løberg, Frederik (2018). "Measuring the Signal Quality of Respiratory Effort Sensors for Sleep Apnea Monitoring: A Metric Based Approach". URL: http://urn.nb.no/URN:NBN:no-65349.

Løberg, Fredrik, Vera Goebel and Thomas Plagemann (2018). "Quantifying the signal quality of low-cost respiratory effort sensors for sleep apnea monitoring". In: *HealthMedia 2018 - Proceedings of the 3rd International Workshop on Multimedia for Personal Health and Health Care, co-located with MM 2018*. Association for Computing Machinery, Inc, pp. 3–11. ISBN: 9781450359825. DOI: 10.1145/3264996.3264998.

Mahanti, Rupa (2019). "Data Quality and Data Quality Dimensions". In: *Software Quality Professional Vol. 22 No. 1*, pp. 4–8. ISSN: 1522-0540.

matplotlib (2018). "matplotlib". Accessed March 28, 2020. URL: https://matplotlib.org/.

Mayo Clinic (2020). "Central sleep apnea: Symptoms and causes". Accessed March 23, 2020. URL: https://www.mayoclinic.org/diseases-conditions/central-sleep-apnea/symptoms-causes/syc-20352109.

McNicholas, Walter (2013). "New Standards and Guidelines for Drivers with Obstructive Sleep Apnoea syndrome". In: SEPTEMBER, pp. 1–49. DOI: 10.13140/RG.2.1.4510.5129.

Medifixit (2016). "Polysomnography (sleep study)". Accessed February 26, 2020. URL: https://www.medifixit.com/blog/polysomnography-sleep-study.

Morgenthaler, Timothy I., Vadim Kagramanov, Viktor Hanak and Paul A. Decker (2006). "Complex sleep apnea syndrome: Is it a unique clinical syndrome?" In: *Sleep* 29.9, pp. 1203–1209. DOI: 10.1093/sleep/29.9.1203.

NOX Medical (2020a). "Nox T3 Sleep Monitor". Accessed February 25, 2020. URL: https://noxmedical.com/products/nox-t3-sleep-monitor/.

numPy (2020). "numPy". Accessed March 28, 2020. URL: https://numpy.org/.

pandas (2018). "pandas". Accessed March 28, 2020. URL: https://pandas.pydata.org/.

Punjabi, Naresh M. (2008). "The epidemiology of adult obstructive sleep apnea". In: *Proceedings of the American Thoracic Society* 5.2, pp. 136–143. DOI: 10.1513/pats.200709-155MG.

pyEDFlib (2018). "pyEDFlib". Accessed March 28, 2020. URL: https://pypi.org/project/pyEDFlib/.

scikit-learn (2018). "scikit-learn". Accessed March 28, 2020. URL: https://scikit-learn.org/.

SciPy (2017). "SciPy". Accessed March 28, 2020. URL: https://www.scipy.org/.

SomnoMed (2020). "Hva er OSA - obstruktiv sovnapne?" Accessed February 23, 2020. URL: https://somnomed.com/nb/pasienter/hva-er-osa-obstruktiv-sovnapne/.

SweetZpot (2020). "Flow". Accessed March 20, 2020. URL: https://www.sweetzpot.com/flow.

The Mathworks, Inc. (2018a). "findpeaks - R2018a". Accessed March 28, 2020. URL: https://se.mathworks.com/help/signal/ref/findpeaks.html.

— (2018b). "matplotlib". Accessed March 28, 2020. URL: https://se.mathworks.com/products/matlab.html.

Traaen, G. M., B. Øverland, L. Aakerøy, T. E. Hunt, C. Bendz, L. Sande, S. Aakhus et al. (2019). "Prevalence, risk factors, and type of sleep apnea in patients with paroxysmal atrial fibrillation". In: *IJC Heart and Vasculature* 26, p. 100447. ISSN: 23529067. DOI: 10.1016/j.ijcha.2019.100447.

Virtanen, Jaakko, Tommi Noponen, Kalle Kotilahti, Juha Virtanen and Risto J. Ilmoniemi (2011). "Accelerometer-based method for correcting signal baseline changes caused by motion artifacts in medical near-infrared spectroscopy". In: *Journal of Biomedical Optics* 16.8, p. 087005. DOI: 10.1117/1.3606576.

Wang, Yequan, Minlie Huang, Xiaoyan Zhu and Li Zhao (2016). "Attention-based LSTM for Aspect-level Sentiment Classification". In: pp. 606–615. DOI: 10.18653/V1/D16-1058.

WebMD (2020). "Apnea Hypopnea Index (AHI)". Accessed March 7, 2020. URL: https://www.webmd.com/sleep-disorders/sleep-apnea/sleep-apnea-ahi-numbers#1.

Wikipedia (2020a). "Random forest". Accessed February 16, 2020. URL: https://en.wikipedia.org/wiki/Random_forest.

— (2020b). "Multilayer perceptron". Accessed February 16, 2020. URL: https://en.wikipedia.org/wiki/Multilayer_perceptron.

— (2020c). "Convolutional neural network". Accessed February 16, 2020. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-4.

— (2020d). "Long short-term memory". Accessed February 16, 2020. URL: https://en.wikipedia.org/wiki/Long_short-term_memory.

— (2020e). "Gated recurrent unit". Accessed February 16, 2020. URL: https://en.wikipedia.org/wiki/Gated_recurrent_unit.

— (2020f). "Cross-validation". Accessed February 17, 2020. URL: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation.

— (2020g). "Cohen's kappa". Accessed February 17, 2020. URL: https://en.wikipedia.org/wiki/Cohen%5C%27s_kappa.

— (2020i). "Deep Learning". Accessed March 24, 2020. URL: https://en.wikipedia.org/wiki/Deep_learning.

— (2020j). "Evaluation of binary classifiers". Accessed April 12, 2020. URL: https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers.

Xu, Liyue, Fang Han, Brendan T. Keenan, Elizabeth Kneeland-Szanto, Han Yan, Xiaosong Dong, Yuan Chang et al. (2017). "Validation of the Nox-T3 portable monitor for diagnosis of obstructive sleep apnea in Chinese adults". In: *Journal of Clinical Sleep Medicine* 13.5, pp. 675–683. ISSN: 15509397. DOI: 10.5664/jcsm.6582.

Young, Terry, Paul E. Peppard and Daniel J. Gottlieb (2002). “Epidemiology of obstructive sleep apnea: A population health perspective”. In: *American Journal of Respiratory and Critical Care Medicine* 165.9, pp. 1217–1239. DOI: 10.1164/rccm.2109080.

Young, Terry, James Skatrud and Paul E. Peppard (2004). “Risk Factors for Obstructive Sleep Apnea in Adults”. In: *Journal of the American Medical Association* 291.16, pp. 2013–2016. DOI: 10.1001/jama.291.16.2013.

Appendices

Appendix A

Source Code

The source code presented and used in this thesis can be found at: https://github.uio.no/morthand/Morten-H-Andersen

Appendix B

Evaluation Results

This appendix includes the signal quality metric scores for all recordings including the corrupt ones. These are presented in Table B.1. For machine learning, the tenfold cross-validation results for the feed forward neural networks CNN and GRU using the different data sizes are shown in Table B.2 and in Table B.3, respectively. The results for the recurrent neural networks using the different data sizes are shown for MLP in Table B.4, for LSTM in Table B.5, for BILSTM in Table B.6, for SBILSTM in Table B.7, and for BIWALSTM in Table B.8. The results for Random Forest are shown in Table B.9.

Recording	**Sensitivity**	**PPV**	**CMP**	**WAPE**
Recording 1	97.46%	98.10%	58.30%	28.03%
Recording 2	96.82%	98.70%	61.54%	13.15%
Recording 3	96.78%	94.64%	48.45%	20.70%
Recording 4	94.41%	77.06%	14.45%	27.35%
Recording 5	98.30%	97.89%	75.81%	16.77%
Recording 6	97.78%	93.37%	64.64%	18.72%
Recording 7	97.75%	93.71%	51.55%	26.31%
Recording 8	96.77%	98.36%	60.00%	14.42%
Recording 9	99.04%	98.01%	73.22%	11.54%
Recording 10	99.41%	97.37%	70.02%	7.71%
Recording 11	96.25%	94.18%	72.15%	19.28%
Recording 12	95.55%	85.92%	39.45%	22.95%
Recording 13	98.70%	92.19%	45.57%	14.47%
Recording 14	98.91%	92.16%	32.33%	12.61%
Recording 15	99.15%	95.34%	54.37%	17.01%
Recording 16	99.26%	87.95%	19.16%	17.10%
Recording 17	98.77%	95.67%	62.08%	11.48%
Recording 18	95.55%	98.54%	68.35%	10.76%
Recording 19	98.51%	96.02%	76.16%	12.04%
Recording 20	99.66%	97.08%	77.92%	12.81%
Recording 21	95.58%	91.88%	51.67%	17.56%

Table B.1 continued from previous page

Recording	Sensitivity	PPV	CMP	WAPE
Recording 22	99.64%	99.20%	91.92%	11.24%
Recording 23	98.94%	99.01%	88.01%	17.04%
Recording 24	98.79%	95.94%	68.89%	22.28%
Recording 25	97.09%	97.32%	68.30%	22.04%
Recording 26	99.17%	97.78%	79.41%	12.86%
Recording 27	99.27%	98.50%	84.74%	10.59%
Recording 28	98.88%	97.13%	74.40%	14.92%
Recording 29	94.45%	90.27%	38.84%	32.17%
Recording 30	90.09%	82.91%	26.23%	29.72%
Recording 31	98.66%	91.41%	57.32%	23.65%
Recording 32	96.44%	95.85%	72.35%	22.28%
Recording 33	99.15%	96.55%	75.52%	15.37%
Recording 34	96.39%	96.57%	57.47%	27.77%
Recording 35	96.99%	81.96%	25.97%	29.03%
Recording 36	99.47%	95.74%	65.75%	12.54%
Recording 37	99.69%	97.73%	82.09%	9.33%
Recording 38	98.95%	97.93%	79.35%	12.49%
Recording 39	98.97%	97.48%	73.89%	13.34%
Recording 40	97.45%	75.89%	21.59%	23.82%
Recording 41	98.98%	97.33%	74.95%	21.21%
Recording 42	98.75%	97.12%	67.05%	13.67%
Recording 43	91.71%	78.26%	3.72%	17.92%
Recording 44	88.58%	87.73%	43.72%	25.16
Recording 45	91.71%	95.99%	37.78%	37.45%
Recording 46	91.47%	97.39%	44.06%	33.59%
Recording 47	96.73%	97.94%	59.72%	26.01%
Recording 48	100.00%	99.08%	87.50%	10.68%
Recording 49	98.96%	97.94%	71.43%	13.56%
Recording 50	97.70%	98.52%	75.81%	6.81%
Recording 51	89.11%	96.10%	56.70%	14.81%
Corrupt 1	78.47%	84.56%	26.22%	19.61%
Corrupt 2	94.83%	80.69%	21.75%	28.59%
Corrupt 3	91.26%	81.19%	28.8%	25.56%
Corrupt 4	87.34%	89.98%	6.67%	48.39%
Corrupt 5	90.92%	77.62%	31.11%	23.48%
Corrupt 6	94.3%	73.39%	10.26%	20.96%

Table B.1: All FLOW signal quality results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW S	72.80%	70.46%	73.68%	0.439
FLOW M	73.32%	71.29%	74.27%	0.452
FLOW L	72.47%	71.48%	72.69%	0.437
FLOW adj. S	75.12%	77.61%	71.58%	0.490
FLOW adj. M	76.09%	78.33%	72.17%	0.505
FLOW adj. L	75.53%	81.06%	69.47%	0.502
NOX S	79.32%	83.09%	74.54%	0.575
NOX M	79.58%	82.32%	75.26%	0.577
NOX L	79.30%	83.84%	73.66%	0.575

Table B.2: CNN 10-cross-validation results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW S	57.30%	40.81%	73.89%	0.141
FLOW M	58.65%	47.15%	70.10%	0.168
FLOW L	58.07%	49.01%	65.17%	0.139
FLOW adj. S	74.70%	78.56%	69.84%	0.481
FLOW adj. M	75.02%	79.97%	69.12%	0.487
FLOW adj. L	75.23%	80.14%	69.03%	0.492
NOX S	78.70%	81.51%	75.09%	0.563
NOX M	79.42%	84.25%	73.71%	0.577
NOX L	77.69%	81.28%	73.19%	0.544

Table B.3: GRU 10-cross-validation results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW S	60.52%	47.27%	73.97%	0.204
FLOW M	62.67%	51.27%	73.66%	0.242
FLOW L	61.59%	51.87%	70.99%	0.222
FLOW adj. S	62.11%	60.14%	64.35%	0.237
FLOW adj. M	62.97%	60.82%	64.89%	0.252
FLOW adj. L	62.63%	61.08%	64.55%	0.250
NOX S	71.39%	61.98%	80.44%	0.417
NOX M	71.50%	62.48%	80.19%	0.420
NOX L	70.77%	63.09%	78.19%	0.407

Table B.4: MLP 10-cross-validation results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW S	51.76%	57.61%	44.36%	0.019
FLOW M	57.24%	48.91%	64.46%	0.132
FLOW L	49.27%	51.28%	50.60%	0.016
FLOW adj. S	74.00%	76.05%	70.69%	0.466
FLOW adj. M	74.40%	78.46%	69.53%	0.475
FLOW adj. L	74.31%	80.12%	68.26%	0.477
NOX S	77.01%	78.44%	74.92%	0.530
NOX M	77.90%	80.23%	74.81%	0.546
NOX L	77.44%	81.04%	72.82%	0.536

Table B.5: LSTM 10-cross-validation results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW S	53.63%	54.24%	52.02%	0.058
FLOW M	56.62%	46.83%	65.10%	0.119
FLOW L	53.16%	71.03%	29.81%	0.010
FLOW adj. S	73.44%	75.48%	70.95%	0.458
FLOW adj. M	73.53%	79.30%	67.46%	0.461
FLOW adj. L	74.15%	78.79%	68.65%	0.471
NOX S	77.63%	79.98%	74.39%	0.540
NOX M	78.33%	82.24%	73.66%	0.555
NOX L	77.68%	80.55%	74.17%	0.543

Table B.6: BILSTM 10-cross-validation results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW M	64.86%	59.85%	69.08%	0.285
FLOW L	60.22%	54.80%	68.06%	0.220
FLOW adj. M	73.49%	80.85%	65.50%	0.458
FLOW adj. L	73.74%	79.11%	67.19%	0.460
NOX M	77.24%	82.09%	72.02%	0.534
NOX L	77.69%	81.73%	72.91%	0.543

Table B.7: SBILSTM 10-cross-validation results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW S	48.80%	50.15%	48.82%	-0.012
FLOW M	51.37%	60.54%	40.69%	0.009
FLOW L	48.73%	55.24%	44.76%	-0.000
FLOW adj. S	75.87%	77.26%	73.71%	0.506
FLOW adj. M	74.46%	75.81%	72.00%	0.476
FLOW adj. L	74.75%	77.40%	70.64%	0.480
NOX S	78.31%	77.8%7	77.63%	0.554
NOX M	78.21%	78.31%	77.17%	0.553
NOX L	77.61%	77.94%	76.16%	0.540

Table B.8: BIWALSTM 10-cross-validation results

	Accuracy	**Sensitivity**	**Specificity**	**Kappa**
FLOW S	49.24%	49.10%	51.13%	0.001
FLOW M	47.40%	52.99%	46.56%	-0.006
FLOW L	49.49%	50.93%	50.73%	0.016
FLOW adj. S	67.67%	65.43%	70.96%	0.352
FLOW adj. M	67.85%	64.43%	71.47%	0.351
FLOW adj. L	68.52%	65.36%	72.80%	0.369
NOX S	67.32%	61.51%	72.73%	0.336
NOX M	67.04%	62.96%	71.46%	0.335
NOX L	67.25%	62.84%	71.63%	0.335

Table B.9: Random Forest 10-cross-validation results