Deep learning to detect obstructive sleep apnea events from breathing

Sondre Hamnvik

Thesis submitted for the degree of
Master in Programming and System Architecture
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2021

Deep learning to detect obstructive sleep apnea events from breathing

Sondre Hamnvik

Deep learning to detect obstructive sleep apnea events from breathing

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Obstructive sleep apnea is a common sleeping disorder estimated to affect nearly one billion adults worldwide. It occurs when soft tissues block the airways during sleep, causing a stop of airflow into the body. Apnea events often occur multiple times per night and can last from ten seconds to over a minute. People who suffer from obstructive sleep apnea often report feelings of fatigue and sleepiness, and the disease has been linked to increased risks of heart disease, stroke, and even traffic accidents. Many patients are undiagnosed and do not receive the treatment they need. A sleep study, polysomnography, needs to be performed to diagnose a person with sleep apnea. It consists of recording the biophysical changes of a patient overnight in a sleep lab and manually examining the gathered data to score every sleep apnea event that occurs. For a patient with severe sleep apnea, the number of apneas is over 30 per hour. This diagnostic is both time-consuming and expensive. A common treatment for obstructive sleep apnea is using a CPAP, a breathing machine that creates positive air pressure in the airways allowing the patient to continue breathing. This master thesis presents a novel approach for detecting obstructive sleep apnea events in real-time by using signal data from only two simple sensors.

Our approach uses Yolo, a fast and robust object detection algorithm that is typically used for detecting objects in images. We generate images of signal data from abdominal and thoracic plethysmograph sensors and detect apnea events in these images. Yolo is well suited for detecting objects with similar features but with varying shapes and sizes. Therefore, it might work well for detecting apnea events of varying lengths and intensities. We have created a model that shows the possibility of using this approach for detecting obstructive sleep apnea. It can be used as a tool to pre-screen possible sufferers of obstructive sleep apnea so that a sleep study is only performed on patients that suffer from obstructive sleep apnea. The tool can also be used as a real-time detection tool, where it was able to detect 87% of apneas that occurred, but it also had almost as many false predictions as it had true predictions. A further possibility of a real-time detection tool like this is treating the apneas as they happen by triggering actuators that disrupt the apneic event.

Contents

Acknowledgements

Firstly, this master thesis would not have been completed without the support and guidance of my supervisor Sagar Sen. His guidance has helped me immeasurably, and his extensive knowledge and determination are the reason that this thesis has been complete. I would also like to thank Pierre Bernabé, Simula, and SINTEF for supporting me in this writing.

Secondly, I would like to thank my family and friends for all the motivation and support I have received.

Chapter 1

Introduction

1.1 Motivation

Obstructive sleep apnea (**OSA**) is a common sleeping disorder that is estimated to affect nearly one billion people worldwide [4] and can have large consequences in the lives of those affected. It involves a reduction or complete stop of oxygen intake during sleep caused by soft tissue in the mouth blocking the upper airway. It is time-consuming and invasive to test for the disorder as you need to record a whole night sleep at a sleep lab with multiple sensors attached to the body to gather all the required data and later analyze it. There have been some attempts at detecting obstructive sleep apnea from the patients own home using only a few sensors combined with machine-learned models, but none have achieved good performance. One of the hypotheses of why they do not work so well is that obstructive sleep apnea can vary significantly in length from 10 seconds to over a minute. We are interested in finding out if a neural network trained for object detection, normally used to detect objects in images, can detect obstructive sleep apnea if the signal data is projected as an image. The reasoning behind why such a model might work is that object detection networks are normally used to detect objects of varying shapes and sizes and may therefore be robust enough to detect apneas of varying sizes from a single sensor.

We are interested in using **Yolo**, a real-time object detection neural network that has shown fast inference and robust performance [5]. This approach has another advantage because of its speed, and that is that it opens up the possibility of detecting sleep apneas in real-time. The potential for this in the future is that there might be a possibility to mitigate the effects of sleep apnea using actuators like nerve stimulation devices that send electric signals to the nerves that control the muscles to keep the airways open [24] without having to use a sleeping apparatus like a CPAP.

1.2 Research questions

The research questions that we have chosen for this project are these:

- RQ1: How does changing hyperparameters influence the performance of an object detection model for detecting OSA?
- RQ2: How well does Yolo work for detecting OSA compared to human annotations?
- RQ3: How does Yolo compare to other machine learning approaches for detecting OSA?

1.3 Approach

To experiment and evaluate our results to answer the research questions, we will need a rigid final solution that evaluates models and compares relevant metrics to see if our model performs well. We will need to compare our results to other approaches using machine learning and the state of the art methods used for detecting apneas.

To accomplish this, we will start the project by reading relevant papers to understand what has been tried before. We have chosen to use the "You only look once" real-time object detection algorithm as our back-end model. Therefore, we will need to be familiar with how this model works before training the models and evaluating them. Therefore, the programming part of this thesis will start by exploring this tool to be familiar with it. The plan is to create a prototype tool that shows that it is possible to detect events in a time series. If this works, we will develop a reproducible model that can be used to detect apneas. We will also develop a way to evaluate the model and a real-time tool to visualize the predictions.

1.4 Results

The experiments done in this thesis have shown promise for using an object detection model for detecting sleep apnea from a combination of abdominal and thoracic sensors. We have explored which features and hyperparameters are important when using this approach, and found that the best results occur when we project 120 seconds of signal data, with the signal being a combination of data from the abdominal and thoracic sensors, with an overlapping stride of 30 seconds between each image generated. This resulted in an MCC-score, which is a score well suited for an imbalanced dataset, of 0.432. Simulating a real-time detector has shown a detector that detected 6679 of the 7644 apneas in the dataset, or 87.38% of all apneas. At the same time, the model detected 6282 apneas that were not annotated as being sleep apnea. This real-time model detected apneas more aggressively than the model with 30-second overlap and had a slightly lower MCC-score of 0.398. We have created a tool for detecting and visualising obstructive sleep apnea in real-time. We have attempted to compare our findings against previous work using machine learning, but because of differences in how the performance is measured, it is difficult to compare models equally. Our model generally has slightly lower scores,

but we predict apneas occurring with decisecond precision, while other approaches use 30 seconds or one-minute precision. We have also learned that training models using images is quite resource-intensive, and we believe that we have only grazed the surface regarding what is possible by using an object detection model for detecting events in one-dimensional data.

A paper from an early iteration of this approach has been presented at the Internation Joint Conference on Artificial Intelligence Demonstrations Track [16].

1.5 Structure of thesis

The thesis is divided into the following Chapters:

Chapter 2, Background. Description of the technical background material needed to understand the approach, including information about apneas and the basics of machine learning. Some information about metrics used for evaluating machine-learned models and earlier approaches attempted similarly to our approach.

Chapter 3, Approach. A chapter describing the tools developed during this thesis to ensure reproducible models and accurate evaluations. It describes the changeable parameters for configuring our model to perform as good as possible, our tool to wrap the model to ensure comparable evaluations, and our real-time tool created to visualize the detection in real-time.

Chapter 4, Evaluation. A chapter describing the research questions and our attempts to answer them. Includes an evaluation of multiple models and parameters and how the model performs. Threats to validity are also included in this chapter.

Chapter 5, Conclusion. Summary of the thesis, summary of our results, critical reflections, and opportunities for future works.

Chapter 2

Background

This chapter will focus on the theoretical information we need in this thesis. In Section 2.1 we will describe sleep apnea and present the different types of apneas, we will describe how a polysomnography is normally performed, and we will explore the dataset we will be using for our project. In Section 2.2 we will provide some background on machine learning basics and details about the implementation we will be using. Section 2.3 will look at the metrics that can quantify the performance of a machine learning model. In Section 2.4 we will look at some of the previous approaches to using machine learning to detect sleep apnea.

2.1 Sleep Apnea

Sleep apnea is a common respiratory disorder occurring when a person has a cessation or large reduction in oxygen intake during sleep. It is estimated to affect nearly 1 billion people worldwide [4]. Many sleep apnea cases are going unnoticed as the symptoms of the disease are difficult to notice, and testing for the disease is expensive. An apnea event is when a person has a reduction in oxygen intake lasting from 10 seconds to over 60 seconds. If this occurs when a person is sleeping, it is known as a sleep apnea event. As the events occur during sleep, it is often difficult to discover that you have symptoms of the disorder yourself. It is often a partner or someone near the vicinity that reports abnormal breathing observations. People may also suspect having sleep apnea based on fatigue when awake as your body does not get the long rest it needs during the night because of the constant arousal's caused by the apnea.

To diagnose somebody with sleep apnea, the patient needs to have a polysomnography performed. A polysomnography (PSG) is a recording of bodily changes during the night that is recorded in a sleep lab or hospital with many sensors attached to the body. The signal data then needs to be manually analyzed and scored by a sleep technician who can finally diagnose the patient. This is expensive and time-consuming. An example of the large amount of data from the multiple sensors a sleep technician needs to look through can be viewed in Figure 2.1.

A person with sleep apnea receives a lower quality of sleep than a

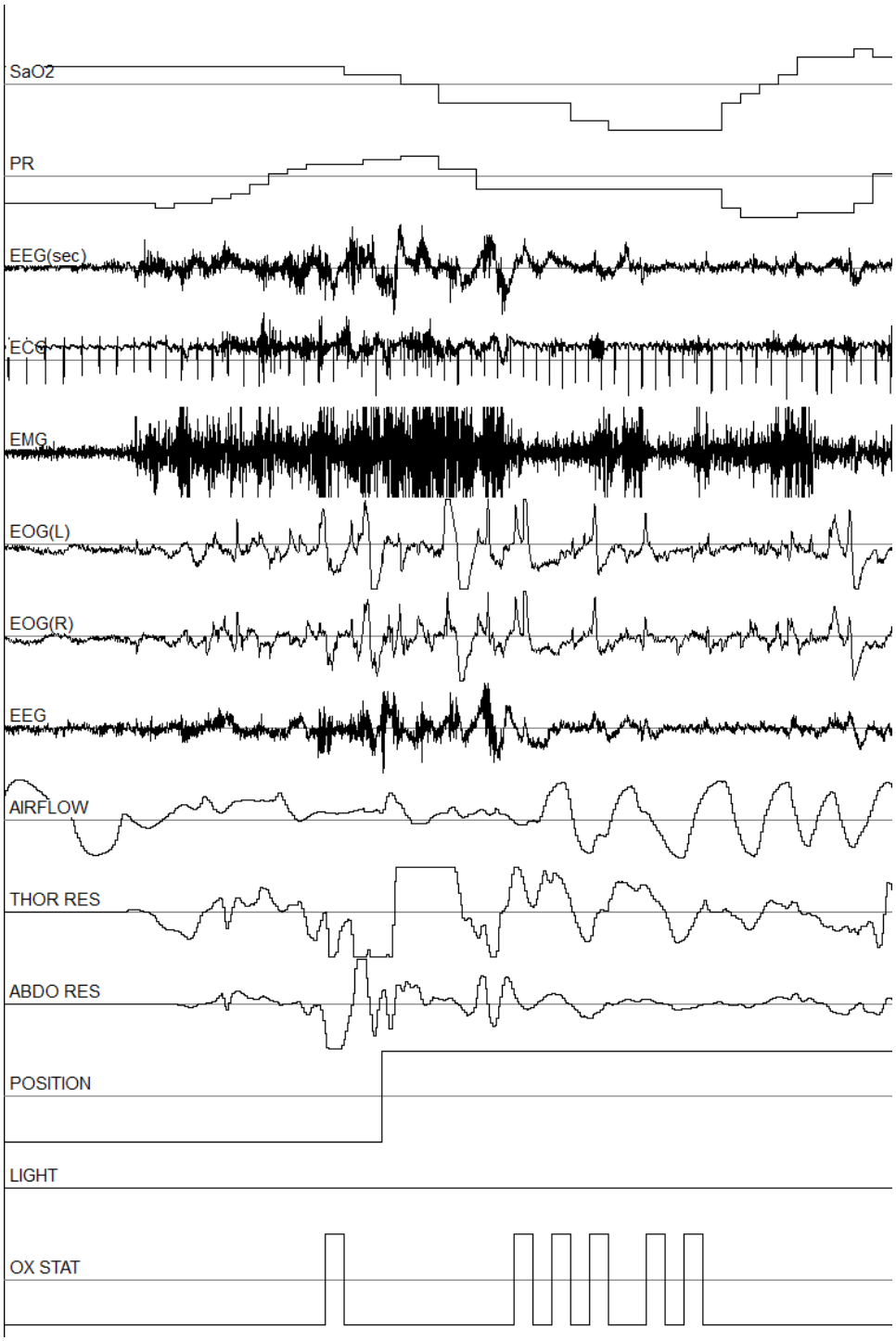


Figure 2.1: Example of multiple signals in a polysomnoagraphy

person without and often feels fatigued when awake. Sleep apnea and the fatigue caused by it has been linked to many other diseases. It can increase the risk of heart disease by 3 times, stroke by 4 times and car accidents by 7 times [14].

2.1.1 Classification of sleep apneas

Sleep apnea is a common intrinsic sleep disorder estimated to occur in nearly 1 billion people worldwide [4]. An apnea is commonly referred to as a reduction or complete cessation of oxygen intake in person for a duration of time. Apneas can further be divided into different categories based on the amount of oxygen reduction occurring and the cause of the cessation. An apnea event is when there is a complete stop of oxygen intake for a person, while a reduction of oxygen intake is classified as hypopnea. We can further divide apneas into obstructive and central sleep apneas based on the cause of the complete stop of oxygen intake. If the apnea is caused by blocking in the airway, it is an obstructive sleep apnea event. If the brain is not sending signals to the body to breathe, it is a central sleep apnea event. A combination of these apnea types is also possible and is called mixed sleep apnea. We will examine these types of apneas further in the following paragraphs.

Obstructive Sleep Apnea

Obstructive sleep apnea (OSA) is caused by blocking in the respiratory tract. Is the most common type of sleep apnea [41]. One common cause is that the tongue blocks the airway in the back of your throat when the tongue relaxes during sleep. A visualization of this form of apnea can be seen in Figure 2.2. When this occurs, there is no oxygen flow into the lungs even though respiratory movements in the abdomen and thorax are maintained. The apnea event is disrupted by the brain arousing the body from a deep sleep so that the patient clears the airway and can inhale. This is often accompanied by loud snores and vocalizations that consists of gasps, moans, and mumbling [41]. A bed partner might also notice the cessation of breathing and might nudge the patient to wake him up out of concern about the stopped breathing. These actions disrupt sleep and cause the patient to feel fatigued when awake. When the patient is awake, they typically feel unrefreshed and may describe feelings of disorientation, grogginess, mental dullness, and incoordination. Sleepiness during the day is a common effect, which can cause incapacitation, resulting in job loss, accidents, self-injury, marital and family problems, and poor school performance [41].

Obstructive sleep apnea syndrome can occur at any age, with the highest incidence occurring between the ages of 40 and 60, with a majority of cases being male. Obesity is often associated with obstructive sleep apnea, but some patients are not overweight, and morbid obesity is present only in a minority of patients [41].

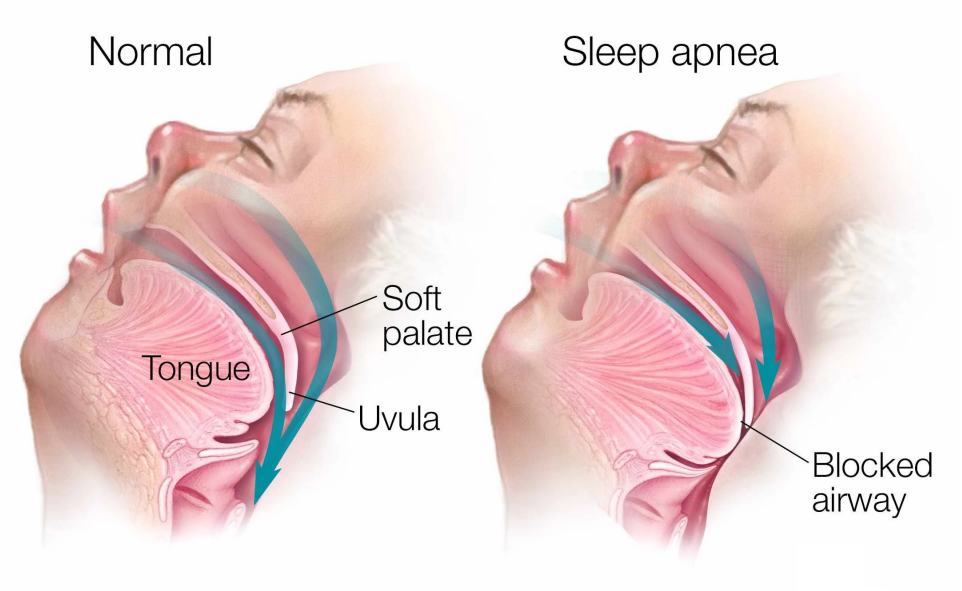


Figure 2.2: Illustration of obstructive sleep apnea[24]

Central Sleep apnea

Another common type of apnea is **Central Sleep Apnea**. This occurs when the brain does not send signals to the body to start the breathing. Therefore the person does not activate its respiratory muscles even though the oxygen levels in the body are dropping below normal levels. Similarly to obstructive sleep apnea, feelings of daytime tiredness, fatigue, and sleepiness are common. A central sleep apnea event typically lasts from 10 to 30 seconds, followed by either gradual or abrupt resumption of respiratory effort. Central sleep apnea is most commonly diagnosed in people over age 65 or people who have neurodegenerative diseases [7].

Mixed Sleep apnea

Patients with central sleep apnea often have respiratory events that consist of both obstructive and central sleep apnea. This is referred to as **Mixed Sleep Apnea**. Many apneic episodes have an initial central component followed by an obstructive component [41].

Hypopnea

Hypopnea is an episode of shallow breathing (airflow reduced by at least 50%) during sleep, lasting 10 seconds or longer, usually associated with a fall in blood oxygen saturation [41]. The symptoms for hypopneas are the same as both obstructive and sleep apneas, with typical symptoms being snoring, fatigue, and difficulties with concentration. In comparison to apneas, a hypopnea event does not fully block the intake of air into the body but limits the amount the body receives. It is combined with the

count of obstructive sleep apneas used for calculating the severity of the diagnosis of a patient.

2.1.2 Polysomnography

A polysomnography, also called a sleep study, is the test used to detect sleeping disorders in a patient. It is performed by recording multiple biophysical changes of a patient sleeping over the duration of a night. An illustration of some sensors attached to the body can be seen in Figure 2.3. It is commonly done in a hospital or a sleep lab. To perform a polysomnography, the patient needs to sleep overnight in a sleeping area with multiple sensors attached to the body and in the room [18]. Figure 2.4 shows five relevant signals and examples of events being classified as obstructive, central, or mixed. A polysomnography usually consists of a minimum of twelve different signals, and usually, more than twenty is used. An example of the number of signals a sleep technician needs to view at the same time can be seen in Figure 2.1 In an article by Ibáñez, Silva and Cauli which summaries current methods for evaluating sleep, we can find an exhaustive list of tests and information gathered during a polysomnography [18].

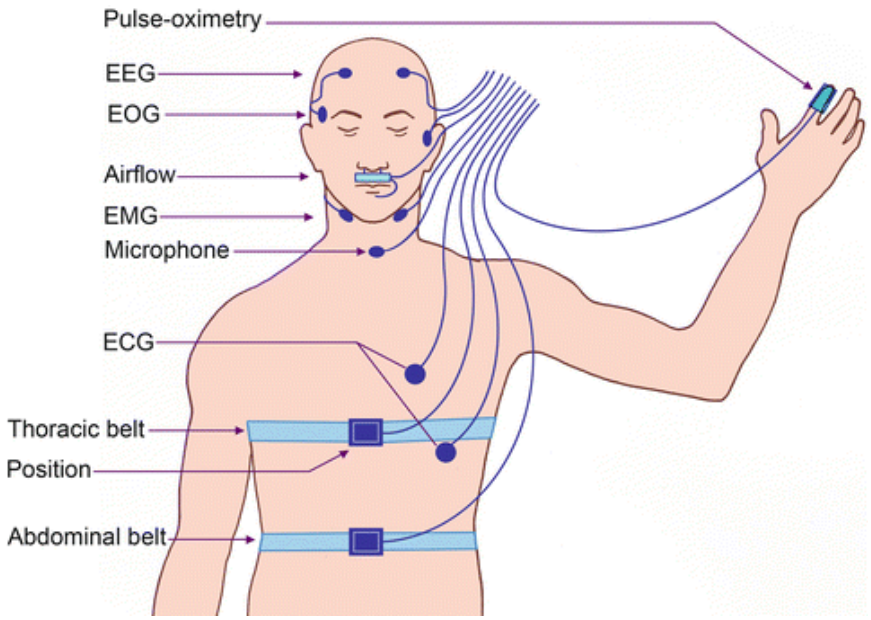


Figure 2.3: Examples of some polysomnography sensors and how they are attached to the body [26].

- *Electroencephalogram (EEG)* – measures and records the brainwave activity to identify sleep stages and detect seizure activity.
- *Electrooculogram (EOG)* — records eye movements. These movements are important for identifying the different sleep stages, especially the REM stage.

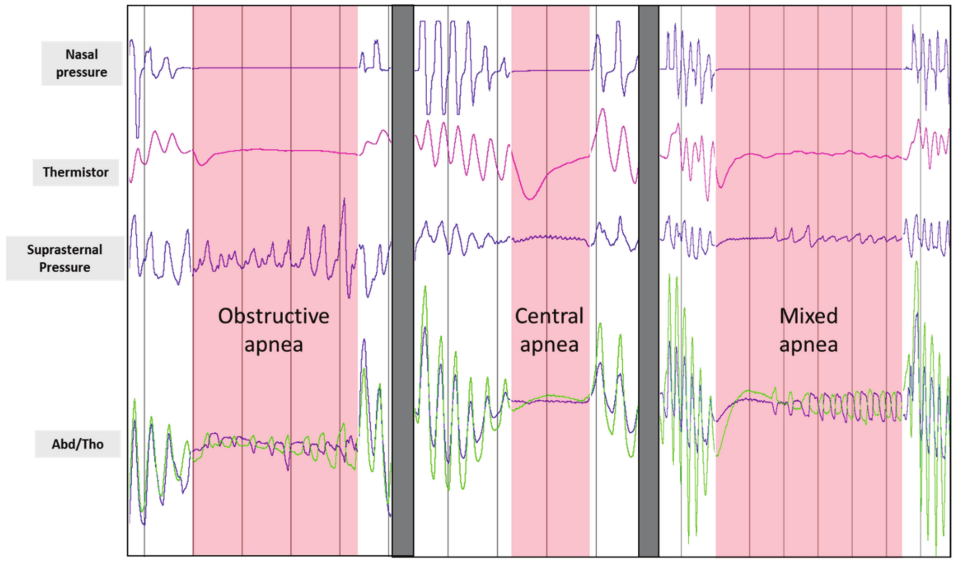


Figure 2.4: Example of obstructive, central, and mixed sleep apnea [13].

- *Electromyogram (EMG)* — records muscle activity (e.g., teeth grinding and face twitches; but also, limb movements using surface EMG monitoring of limb muscles, periodic or other). Chin EMG is necessary to differentiate REM from wakefulness. Limb EMG can identify periodic limb movements during sleep (PLMS).
- *Electrocardiogram (EKG)* — records the heart rate and rhythm.
- *Pulse oximetry* — monitors oxygen saturation (SO2)
- *Respiratory monitor* — measures the respiratory effort (thoracic and abdominal). It can be of several types, including impedance, inductance, strain gauges, etc.
- *Capnography* — measures and graphically displays the inhaled and exhaled CO2 concentrations at the airway opening.
- *Transcutaneous monitors* — —measure the diffusion of O2 and CO2 through the skin
- *Microphone* — continuously records the snoring volume and kind
- *Video camera* — continuously records video. It is useful to identify the body motion and position.
- *Thermometer* — records the core body temperature and its changes.
- *Light intensity tolerance test* — determines the influence of light intensity on sleep
- *Nocturnal penile tumescence test* — is used to identify physiological erectile dysfunctions

- *Esophageal tests* — includes pressure manometry, to measure pleural pressure; esophageal manometry to assess peristalsis and esophageal pH monitoring (acidity test).
- *Nasal and oral airflow sensor* — records the airflow and the breathing rate.
- *Gastroesophageal monitor* — is used to detect Gastroesophageal Reflux Disease (GERD)
- *Blood pressure monitor* — measures the blood pressure and its changes

A polysomnography is the most advanced tool used to diagnose many sleep disorders. One of the reasons is that it uses advanced tools that are extremely precise, which in turn allows a sleep technician to apply his knowledge to detect apneic events in the recording. Polysomnography is considered the *gold standard* for detecting sleep apnea. The downsides of detecting sleep apnea using polysomnography is that it is time-consuming to record, and it requires trained professionals to annotate the data. This is why the polysomnography is costly to perform, and it is common to only record a single night's sleep for analysis. Another downside is that the patient is sleeping away from home, which might cause the sleep to be different from normal.

After the night has been fully recorded, a sleep technician needs to analyze the gathered data to annotate the apnea events. The recording is analyzed and annotated according to the International Classification of Sleep Disorders, [33]. To classify events, a sleep technician needs to look at patterns in the sensor data to try to understand what is happening in the body. A report from a sleep study can be as many as five pages long [11] and it might take up to two weeks until the scoring is complete [44]. Before the recording, the technician usually spends around 45 minutes completing the hook-up of all electrodes and sensors [17]. The fact that the recording and scoring is time consuming combined with the space requirement of each patient needing his or her own bed and room is a large barrier in testing for sleep apnea. One of the values reported in the report generated from polysomnography is the apnea-hypopnea index.

Apnea-Hypopnea Index. The Apnea-Hypopnea Index (**AHI**) is a calculated value used to indicate the severity of sleep apnea a patient has. Equation 2.1 shows the equation for calculating the Apnea-Hypopnea Index.

$$\text{AHI} = \frac{\text{Apneas + Hypopneas}}{\text{Hours of sleep}} \times 100 \tag{2.1}$$

This Apnea-Hypopnea Index is calculated by adding the number of distinct apneas occurring during sleep with the number of distinct hypopneas occurring during sleep, and then dividing this value by the number of hours of sleep. The index represents how many apneas and hypopneas occur on average per hour. An AHI value of less than 5 is considered normal sleep. An AHI value between 5 and 15 is categorized

as mild sleep apnea, a value between 15 and 30 is moderate sleep apnea, and more than 30 is considered severe sleep apnea.

2.1.3 Treatment

A continuous positive airway pressure machine (CPAP) is one common treatment for managing obstructive sleep apnea. It is a machine that causes constant positive air pressure in the upper respiratory tract, hindering the obstruction occurring. The use of CPAP is not without its drawbacks. Common complaints after use is general discomfort, nasal congestion, abdominal bloating, mask leaks, claustrophobia and inconvenience of regular usage [28]. The adherence to the use of CPAP for treatment is often low. Eight to fifteen percent of patients stopped using the treatment after a single night use, and within one year, up to 50% of patients stopped using the CPAP [3].

A lifestyle change might help treat obstructive sleep apnea in mild cases. Some other recommendations are losing weight, exercising, quitting smoking, not sleeping on your back, not drinking alcohol, and not using sedative medications [24]. There are also some other approaches to treating obstructive sleep apnea, like surgically removing tissue, jaw surgery, implants, and mouthpieces that open up the airways[24].

2.1.4 Sleep Heart Health Study

The Sleep Heart Health Study (SHHS) is a multi-centre cohort study conducted by the American National Heart Lung & Blood Institute [40]. It is a study that consists of sleep recordings from two distinct date ranges. The first recording range is between September 1995 to January 1998 and will be referred to as SHHS-1. The second recording range is between January 2001 to June 2003 and will be referred to as SHHS-2 and is a follow up on a subset of patients from SHHS-1.

SHHS-1 consists of sleep data from 6441 individuals selected from a collection of patients that had already participated in one of nine existing epidemiological studies. The Sleep Heart Health Study invited patients that fulfilled the following criteria:

- Age is 40 years or older
- No history of treatment of sleep apnea
- No tracheostomy
- No current home oxygen therapy

Among the patients that fulfilled these criteria, 6441 individuals were enrolled in the SHHS-1 study, and of these individuals, 5804 patients are available in the dataset. The number of individuals is reduced due to data sharing rules on certain cohorts and subjects [37]. The SHHS-2 study consists of recorded data from a follow-up clinic visit from 4080 patients from the SHHS-1 study.

The dataset from the study consists of sleep recordings of patients recorded from a total of 19 sensors with a sample rate occurring between 125 Hz and 1 Hz [39]. Each patient has a recording from one night at a minimum. Associated with each recording is an annotation file scored by a sleep technician. This file contains information about events scored during the recording.

Signals

The signals included in the Sleep Heart Health study are signals from EEG, EOG, thermister, chin EMG, ECG placement, respiratory bands, position, light, and oximetry [39]. SHHS-1 contains 17 distinct signals, and SHHS-2 contains 16 distinct signals. All signals were recorded using Compumedics P-Series Sleep Monitoring System consisting of a main unit and a patient interface box.

This master thesis will focus on the abdominal and thoracic expansion and contraction signals which are available in both SHHS-1 and SHHS-2. Respiratory bands are used for measuring, and both use the Respitrace Inductance Plethysmography sensor to detect the volume changes [39]. The dataset has two signals labelled "abdo_res" and "thor_res", referring to these sensors. The signals are sampled at every decisecond (sample rate of 10 Hz). The signals have been normalized and the values range between minus one and plus one.

Annotations

The recorded signals have been annotated and scored by sleep technicians. The findings from the sleep technicians scoring are stored in .xml files in the dataset and contain the events they have scored. They also record and store the sleep stages of the patient.

Every sleep recording is broken into a hypnogram representing the different sleep stages during a recording. These sleep stages can be: Awake, sleep stage one to four, and REM sleep. These stages are recorded every 30 seconds and are represented with an integer between and including zero and five. A value of zero is awake, one to 4 is sleep stages, and five is REM sleep.

The annotation file also usually contains multiple scored events. Each event has an event name (f.ex Obstructive Apnea), the start time in seconds since the start of recording, the duration in seconds, and the input sensor the scoring is tied to. The information about the start of the recording and the duration is stored as a positive floating point value. The input sensor information is scored as a string. Optionally, scoring may also contain information about the lowest Sp02 and the desaturation level in this duration. Examples of scored events are Sp02 desaturation, obstructive apnea, hypopnea, arousal, Sp02 artefacts, and central apnea.

The annotations are stored in the dataset in two different file formats containing the same data. One is the XML annotation exported from Compumedics Profusion [38], The other is stored as XML-NSRR. Both

```
...
<ScoredEvent>
    <EventConcept>Recording Start Time</EventConcept>
    <Start>0</Start>
    <Duration>30600.0</Duration>
    <ClockTime>00.00.00 23.00.00</ClockTime>
</ScoredEvent>
...
<ScoredEvent>
    <EventType>Respiratory|Respiratory</EventType>
    <EventConcept>
        Obstructive apnea|Obstructive Apnea
    </EventConcept>
    <Start>7898.9</Start>
    <Duration>15.4</Duration>
    <SignalLocation>ABDO RES</SignalLocation>
</ScoredEvent>
...
```

Figure 2.5: Example of parts of XML file with the first scored informing us that the recording started 23.00 and lasted for 30600 seconds (8,5 hours). The second scored event is a scoring of Obstructive apnea, starting after 7898,9 seconds after the start of the recording, and lasted for 15,4 seconds.

formats allow for easy extraction of scored events and sleep stages, and we have chosen to use XML-NSRR in this thesis. Figure 2.5 shows an extract from an XML-NSRR annotation file.

Dataset characteristics

The total dataset is around 374 gigabytes in size. SHHS-1 consist of 5804 patients with an average age of 63.1 years old, with a standard deviation of 11.2 years. The minimum and maximum ages in this dataset is 39 years and 90 years. 52.3% of patients were female. The average AHI value was 9.6 [8].

2.2 Machine learning

Machine learning is the practice of making computers modify or adapt their actions to get more accurate, where accuracy is measured by how well the chosen actions reflect the correct ones [21]. This section will present multiple paradigms within machine learning, some network architectures commonly used and their use cases, how object detection works in machine learning, and finally, some information about the machine learning process.

2.2.1 Machine Learning Paradigms

Machine learning is commonly divided into three categories. It is also not uncommon to combine parts of these different approaches. The main categories are as follows:

Supervised learning

Supervised learning is machine learning where the algorithm has some data points as input with the desired output associated with each data point, commonly called a label, ground truth, or class. When a machine learning model learns using the supervised learning paradigm, it takes data points from a training dataset, runs it through an neural network, and calculates a prediction of the same type as the label. Typically, predictions are a class name or a value. The model can then calculate how far away from the true label the prediction was and use this error information to recalculate the model. To avoid the model changing too much at every step in the learning process, it is common to group multiple data points and annotations into a batch. By calculating a change to the model that returns a better score for most of the data points in the batch, we can recalculate the model and make it more applicable for generalization.

Unsupervised learning

Unsupervised learning is also machine learning on data, but in comparison to supervised learning, we don't have (or use) labels noting the true value of our data points. The main reason for using unsupervised learning is that it helps to find clusters and features in the data automatically. It may discover some similarities between each input vector that humans would not recognize. Since labelling of data is quite expensive, it may also be more cost-effective to use unsupervised learning than supervised learning, as it does not need the same ground truth as supervised learning does. Unsupervised learning is often used for image recognition, spam detection, and analysis of customer segments [46].

Reinforcement learning

Reinforcement learning is similar to unsupervised learning in that it does not use ground truth labels to evaluate performance. In reinforcement learning, you create an environment to train in and an actor capable of performing actions. When an actor performs single or multiple actions, it receives a reward if the outcome of the action is a preferable outcome. After many iterations, the actor can explore and try new things and try to find new solutions and approaches to challenges to better perform in the environment. In real life, it has been used for finding solutions for elevator scheduling, robot movement, and logistics, among other things.

2.2.2 Neural Networks

Neural networks can be explained as a series of algorithms that take input data, processes it, and outputs some predictions. A neural network is a combination of algorithms built in a way to emulate the human brain [9]. The building blocks for these algorithms are neurons, which are a mathematical model inspired by neurons in the brain. In Figure 2.6 we see a model of a single neuron. The inputs x_i are multiplied with the weights w_i, and the neuron sums the values. It runs through an activation function which may be a simple threshold function that decides whether the neuron fires or not for the current input if it is a binary neuron. If the neuron is not a binary neuron, it may output a floating-point or integer value [21]. This subsection will describe the Perceptron, a simple network and a building block and precursor to many other neural networks and convolutional neural networks commonly used for object detection.

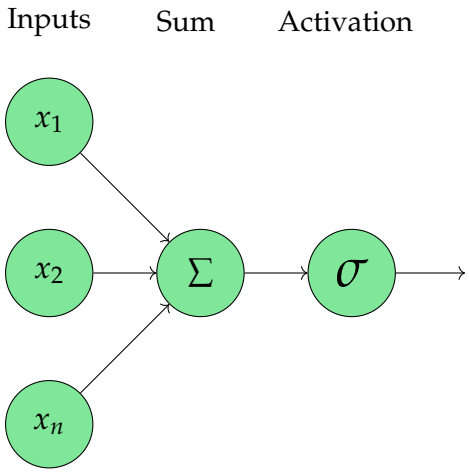


Figure 2.6: A model of a neuron

A neural network consists of multiple connected neurons performing mathematical operations to output a prediction. By stacking these neurons after each other in different configurations, we can generate networks that work great for different types of tasks. In Figure 2.7 we see an example of a simple Neural Network with an input layer created out of two features, a hidden layer with four neurons and a single output layer.

Perceptron

The **Perceptron** is one of the simplest forms of a neural network and is a great example for understanding the basics of how a neural network is created. The perceptron can be created using only a single neuron or a collection of multiple neurons. Each of these neurons are independent of each other. If a neuron's input depends on another neuron, we have created a multi-layer perceptron and have started implementing a more advanced network. Perceptrons are quite similar to a single neuron. The

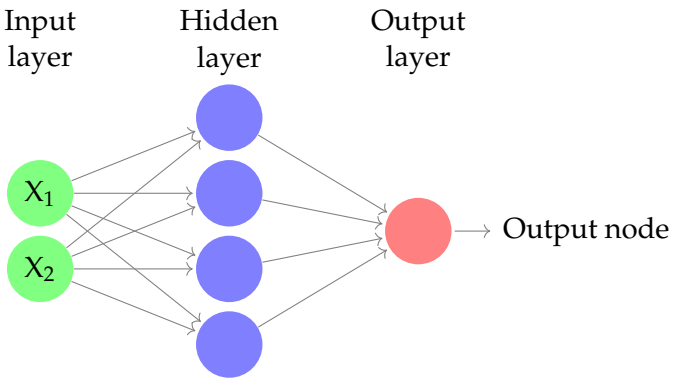


Figure 2.7: A simple network consisting of three layers [9]

main difference between a perceptron and a single neuron is the possibility of using multiple neurons in the network and weights for each input to the neuron. These weights can be updated in a training stage so that the model learns to perform better.

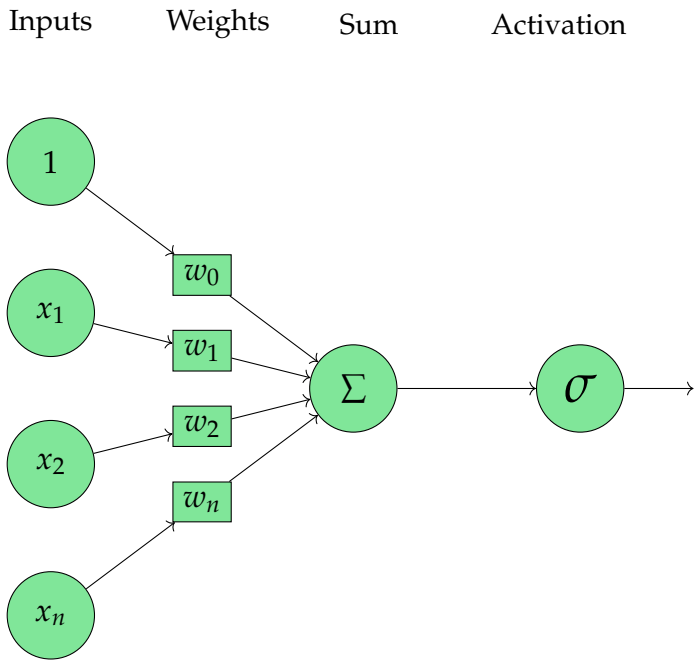


Figure 2.8: A perceptron

Figure 2.8 shows a perceptron with n inputs. The first input is a bias; its function is a value that can be used to shift the activation function to the left or right. Each of the other inputs is multiplied with a weight and summed together. The summed value is then run through an activation function that modifies the value somehow. If the value is not as expected, we will need to adjust the weights until we get a better result, or the data

might not be linearly separable.

Convolutional neural networks

A convolutional neural network is a type of neural network often applied for image recognition. It is more advanced than the perceptron. In its simplest form is based on a multilayer perceptron with a few regulations to avoid overfitting the model to the training data. Something which would give worse performance when applied in the real world. The Neocognitron proposed by Fukushima is one of the first types of convolutional neural networks and introduced the two basic layers in a convolutional neural network: the downsampling layer and the convolutional layer. A convolutional layer aims to detect features and feed them to the next layer in the network. In convolutional neural networks, you often have small filters that convolve over the whole input image and detects shapes, edges, colours, and gradients. An abstraction of a convolutional neural network can be seen in Figure 2.9. A combination of many filters gets fed to the next layer in the network, which uses its own filters to look for patterns. Since these are hidden layers, it is difficult to give examples of what they are detecting.

A downsampling layer is a layer that reduces the input dimension from the previous layer to the next. It combines a cluster of neurons into a single neuron, of which the output can be fed into the next convolutional layer. Using a downsampling layer, we can move from detecting small features in the images to detecting larger objects in a more efficient way than checking a large number of small features. A downsampling layer is also known as a pooling layer.

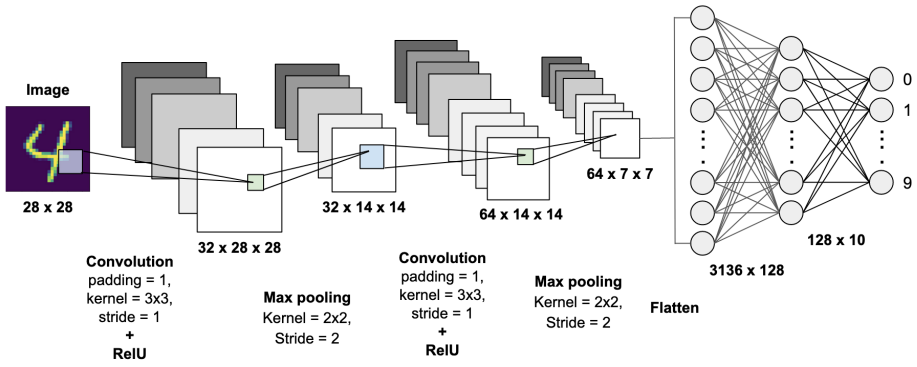


Figure 2.9: A convolutional neural network illustration [36]

2.2.3 Object detection

Object detection is a topic in computer science within the subsection of computer vision. The main goal of an object detection algorithm is to identify and classify objects in an image. This is quite a difficult task as the visual features of an object can vary substantially in an image in different

situations. An object can look different depending on things like light and shadows, proximity to a camera, rotation, and occlusion behind other objects. An object of a class can also look quite different from other objects in the same class. If we have a class called "bird", we should classify both an eagle and a flamingo as the same "bird" class, even though they appear very different visually. There are many implementations of object detection networks, many of which build on convolutional neural networks. In this thesis, we will use the You Only Look Once algorithm, which is a fully convolutional neural network. A fully convolutional neural network is a convolutional neural network that can take an input of arbitrary size.

You Only Look Once

You Only Look Once (**Yolo**) is an approach to object detection first presented in 2015 by Redmon et al. [31]. Redmon et al. reframes object detection as a regression problem to spatially separate bounding boxes and associated class probabilities. By making the detection pipeline as a single network, it can be optimized and perform very fast. Compared to other object detection algorithms, it only needs a single pass through an image to perform all detections. Other algorithms usually examine several regions in images to find objects, while Yolo reasons about the whole image and predicts multiple boxes for each object.

Figure 2.10 shows examples of using Yolo for predicting objects in images. It can reason globally about the whole picture when making predictions[31]. Yolo can be very specific if trained on a large and well annotated dataset. The images in this figure show its capabilities for discovering small objects, overlapping objects, and objects with small differences like three different types of dogs, even discerning photos of malamutes and husky's which looks quite similar. As seen in the figure, Yolo predicts bounding boxes for each prediction.

Yolo divides each image into S x S regions. If an object has a centre in one of these regions, then this cell in the S x S grid is responsible for predicting boxes of the object's boundaries and the confidence level of this bounding box being correct for this class. Unifying the prediction in this way allows Yolo to perform object detection with high performance, with only a small reduction of accuracy. The architecture of Yolo can be seen in Figure 2.11.

Yolo has further been improved and is currently at version 4 [5]. Bochkovskiy, Wang and Liao are the current developers for Yolo and have improved the model further with their version 4. Figure 2.12 shows a comparison of Yolo's performance in image detection against other neural networks. The x-axis represents how many predictions can be done per second, and the y-axis represents the average precision on the dataset. We can see that Yolo performs slightly worse than the best other neural networks, but a lot faster. The best models can predict around 5 images per second, while Yolo can predict over 30 per second with only a small penalty in accuracy. This is good for us as this allows for a faster evaluation.

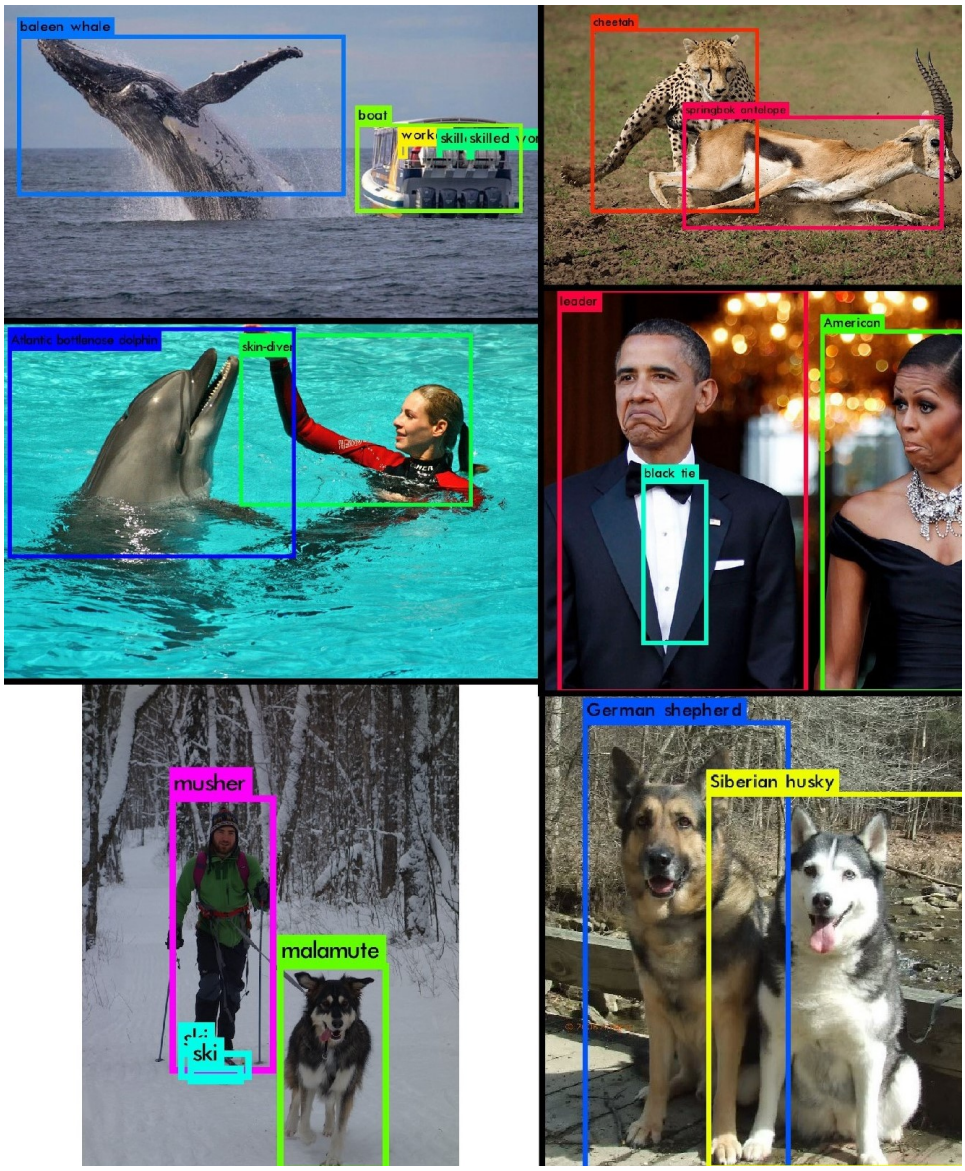


Figure 2.10: Examples of Yolo used for detecting animals, objects, and people [30].

2.2.4 Machine Learning Process

Humans have naturally trained on detecting objects our whole life, and we are quite adept at this task. A computer starts without any knowledge of the world and needs to learn everything from scratch. To train a model that performs well, there are many considerations to be had, and there are some common steps in the process that should be followed. In this section, we will be explaining some important steps in a machine learning process.

Data collection. Data collection is the first step when training a model. A new type of model is often trained and evaluated on an existing dataset.

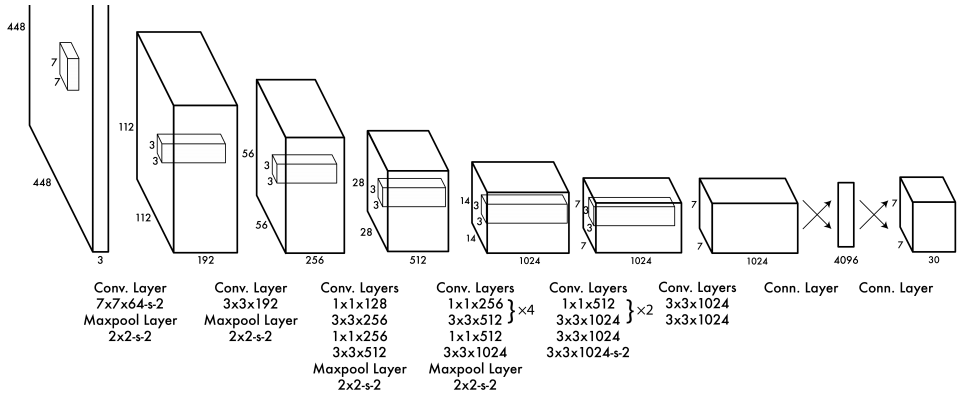


Figure 2.11: The architecture of Yolo [31].

This allows the developer to compare the new algorithm to the results of others. If we are learning about a new problem, we might have to collect the data from scratch or assemble and prepare the data from various sources [21]. Often, large amounts of available data might be relevant but can be hard to use as it exists in many places and formats, and merging it is difficult. It might also be a challenge to ensure that the data is clean without large errors or missing data.

Generally, a larger dataset will result in a better and more robust trained model than a model trained on a small dataset. One of the most common datasets for image recognition has 328 000 images and 2,5 million labelled instances for 91 object types [20]. A model generally performs better with a larger collection of good data than a small one, but the cost is more time spent on computing.

Featurisation When we have our data collected, we can extract the features we need and discard the unneeded features. If we are doing something novel that needs data collection, we might have done this already if we only collected the data we need. We might also apply algorithms to the data to scale it or combine it to generate new features if we believe this might affect the next steps in the machine learning process.

Algorithm choice With the data collected and featurized, we can select the appropriate algorithms for learning on this data. We can choose to use algorithms that have previously been used for similar data or problems, or we may try to do something novel. Some algorithms are good for detecting sequences of data. Some are great for analyzing and discovering clusters and segments of data. Some are good for detecting objects or calculating values from new data.

Most of the algorithms have parameters that can be tweaked. Finding good parameters can be difficult, but you can learn what other approaches have tried, or you can try doing experiments and comparing the results you get from the multiple parameter settings. This might be quite computationally expensive if you are training on a large dataset or with

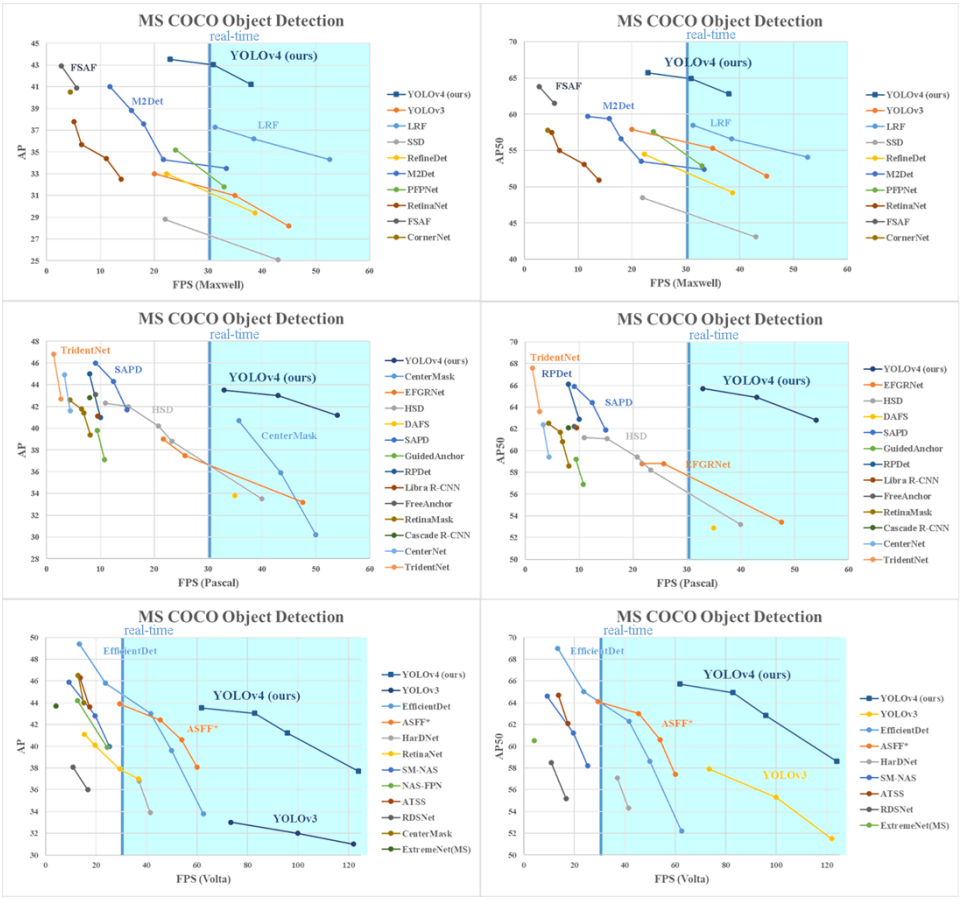


Figure 2.12: Comparison of speed and accuracy of different object detectors [5]

a slow algorithm. Still, it is an important part of configuring the model to perform well.

Training, validation and test sets. It is recommended to split the data we have into multiple sets and only train on one part of the data. If we were to train a model on all the available data we have, we might experience something called overfitting. This happens when the model has seen every available training point and optimized its algorithm to perform well on this. When using the model in the real world later, it will see new data, and it will probably not perform as well as it did on the training data because it is expecting only the training data.

To ensure that the model can work as a general model, we often train on only part of the dataset. We create a training dataset with the data we wish the model to train on, a validation used intermittently to check for overfitting, and a test set that is used for the final evaluation of the model. If the model starts performing worse on the validation dataset, we can assume that we have started to overfit our model and should stop the training. The test set is kept away from all training to make sure that we

are evaluating only unseen data.

Training After doing the previous stages, we can train the model. We need to choose how long we want to train the model for. Typically, we choose to train a model for a set number of batches or train until the average performance gain after a batch is below a certain threshold. Stopping the training when the performance on the validation dataset starts getting worse is also a way to stop the training. For advanced models, it is common to train using a graphical processing unit, which is very effective at doing calculations in parallel. For simpler algorithms, a computational processing unit might perform fast enough.

Evaluation We evaluate the model we have trained to see how well it performs. We choose some metrics to measure how well the model performs, and then we can evaluate using our model on our test dataset. If testing on data without labels, it can also be evaluated by comparing to a human expert who evaluates the predictions. In the next section, we will look at some metrics that are commonly used to evaluate a model's performance.

2.3 Measuring a models performance

If we are not using an expert for manually evaluating the model's performance, it is common to select a representative metric for evaluating the model. When choosing a metric, it is important to keep in mind the type of data we are evaluating and what a good model should perform well at. Especially in medicinal use and in others, it is important to think about two types of errors. Type 1 errors are false-positive conclusions, while type 2 errors are false negative conclusions. The consequences of classifying wrong can have big implications and need to be kept in mind when choosing a metric. In this section, we will present multiple metrics that can be used to quantify the model's performance and the pros and cons of the metrics.

2.3.1 Confusion Matrix

The confusion matrix is a two-dimensional matrix with all possible classes in the horizontal and the vertical axis, with one axis representing the predicted class and the other representing the true class. An example of a confusion matrix for a binary predictor can be seen in Table 2.1 [21]. Here we see that when we predict a class to be true, and the actual class is true, we get a true positive. Similarly, we have false positives, false negatives and true negatives in the other cells.

In a perfect classifier, we want to have all the predictions fit in either the true positive or true negative cells. The values from the confusion matrix are often used in other metrics such as recall, accuracy and precision.

		Predicted	
		Positive	Negative
True	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Table 2.1: Confusion matrix of a binary detector

2.3.2 Metrics

Accuracy. Accuracy is one of the quantifications of how close we are to true values. Equation 2.2 shows the formula for calculating accuracy. It tells us how many correct predictions we made as a fraction of all possible predictions. A disadvantage with using accuracy is when the dataset is highly imbalanced. If the data consisted of 95% of class A, and the remaining 5% was of class B, then a classifier that predicted everything to be class A would have an accuracy score of 0.95, which might be misleading regarding the performance of the classifier.

$$\text{Accuracy} = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} \tag{2.2}$$

Precision. The precision value tells us how relevant each prediction we make is, i.e. if we predict something, how likely is it that the prediction was correct. Equation 2.3 shows the formula for calculating precision. The precision is the ratio of true positive detections against the true positive and the false positive detections. The precision does not give any information about how many predictions were not predicted.

$$\text{Precision} = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2.3}$$

Recall. Recall is the metric that tells us the ratio of true predicted classes predicted against the total number of true classes. Equation 2.4 shows the formula for calculating recall. A high precision value tells us how good we are at detecting all the true classes but ignores the number of predictions.

$$\text{Recall} = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.4}$$

F1 score. The F1 score is a metric that balances the recall and precision values. Precision tells us the ratio of correct prediction vs the total predicted positives. The recall tells us the ratio of true positive predictions results divided by all the true positive truths. The best value the f1-score can have is one, indicating perfect precision and recall, and the lowest is zero if either precision or recall is zero. Equation 2.4 shows the formula for calculating the f1-score.

$$\text{F1} = 2 * \frac{Precision * Recall}{Precision * Recall} \tag{2.5}$$

ROC. The receiver operating characteristic curve (ROC curve) is not a metric but a graphical plot that illustrates the ratio of precision and recall when a minimum confidence threshold is changed in the classifier. It plots the true positive rate against the false-positive rate,

AUC. The area under the ROC curve (**AUC)** measures the area under the ROC curve. It is a metric that aggregates the classifier's performance in all thresholds. A perfect classifier has an AUC of one. It can be used to figure out which threshold your classifier should use if you were to value precision and recall differently.

MCC. Matthews correlation coefficient (**MCC**), or a phi coefficient, is a metric for evaluating a binary classifier that works well for unbalanced datasets. It is a correlation coefficient between the observed and predicted binary classifier. An MCC-score of one means a perfect prediction, a zero is random predictions, and minus one means predictions and ground truth differ perfectly. Equation 2.6 shows the formula for calculating MCC.

The reason for using this metric is that it is not sensitive to data imbalance [6]. It has, together with AUC, been chosen as a best practice metric for validation of predictive models for personalized medicine by a study employed by the United States Food and Drug Administration. [35]

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.6}$$

2.4 Earlier machine learning approaches to detect Obstructive Sleep Apnea

The consequence of undiagnosed sleep apnea, combined with the difficulties of testing, is a great motivator for developing reliable, quick ways to test for it. The hope is that a simpler test can make sure more people can get the treatment they need to treat sleep apnea. There are automated tools in use for helping predict sleep apnea events for the technicians. However, the technician still needs to spend time looking through the whole dataset to see that the classifications are correct, and the polysomnography needs to be recorded correctly.

There have been many different approaches towards using machine learning models for apnea detection without needing to do a full polysomnography. Many of these approaches have been quite successful, but there is still room for improvement to detect apneas with more precision using less invasive sensors. The optimal model should be able to predict all apneas and hypopneas perfectly. In Section 2.1 we mentioned AHI as a value that represents the severity of sleep apnea in a patient. A machine

learning model could also have value if it can predict the patient's AHI value. As the gold standard in detecting apnea is the expensive and time-consuming polysomnography, it would be good to do some prescreening of severity. This is so that the polysomnography analysis will mainly be done on patients with a high percentage of having sleep apnea, utilizing the sleep technicians on patients with a high chance of suffering from sleep apnea. It is also important that we predict few false negatives, as the consequences can be quite large for patients if they go undiagnosed. In Table 2.2 we can look at a few selected papers of earlier approaches, which classifiers they used, and the results that each classifier reported. We can also see from Thorey et al. (2019) who compared results from 5 different sleep technicians annotating the same data, and see that the human experts reached an average inter-scorer accuracy of 75% [43].

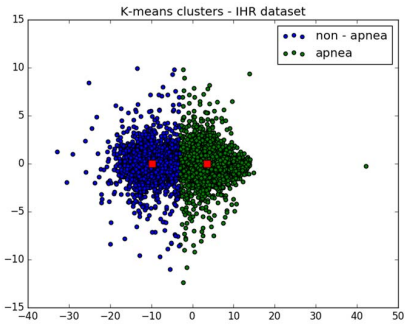
Table 2.2 shows that there are several different approaches towards detecting apnea from sleep data. They have attempted to detect obstructive sleep apnea using different sensors, multiple different classifiers, and population sizes varying from twenty to one hundred. We have also included some of the metrics they have been used for evaluating and can see that there is no single metric used by all papers. The closest is the accuracy, which we in Section 2.3 discussed as a measurement that's not very good for an imbalanced dataset. The papers referenced attempt to tackle the problem of a full polysomnography being expensive and time-consuming and propose ways to test from home using fewer sensors.

Haidar, Koprinska and Jeffries informs us that the current state of the art method when the paper was written in 2017 was using support vector machines. They used f1-scores for evaluating their model. Using support vector machines, they reached an f1-score of 0,72, and by using a convolutional neural network Haidar, Koprinska and Jeffries managed to reach an f1-score of 0,746. They hypothesise that the convolutional neural network can automatically learn statistical features from data, while a support vector machine needs manually extracted features. When humans are satisfied with the extracted features, there will be no more optimisation of choices. At the same time, a convolutional neural network can continue to improve to try to find better solutions. Their approach to predicting the apneas is done by splitting the dataset up into 30 second segments without overlap, that are labeled as either being normal breathing, or abnormal breathing. The dataset they use have been balanced with 12240 abnormal segments, and 12240 normal segments.

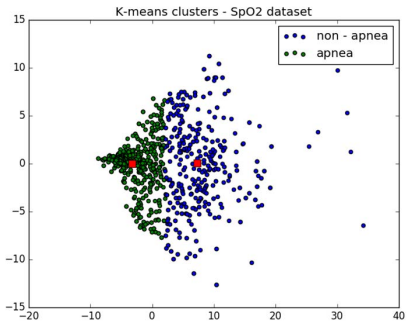
Pathinarupothi et al. uses sensors for blood oxygen saturation and instantaneous heart rate for predicting apnea. Based on earlier information, they suspect a dependence of instantaneous heart value based on obstructive sleep apnea condition and a dependence of blood oxygen saturation based on the obstructive sleep apnea condition. To explore this, they convert the recordings into vectors and save the classification of either apnea or non-apnea. They then group them using k-means clustering and PCA, which are algorithms for grouping data and reducing dimensionality. They then see that the centroids of both classes are far apart, which informs the researchers that there is a dependency. Visualisation of this can be seen

Paper	Pop.size	Sensors	Classifer	Sen	Spc	Prec	Acc	Other
[15]	100	Nasal airflow	CNN	74,7		74,5	74,7	f1=74,6
			SVM	72		72	72	f1=72
[14]	20	3D supernastral accelerometer	CNN					r=0,79
			LSTM					r=0,76
			CNN+LSTM					r=0,84
[19]	69	3D face scan	CNN				68,75%	
[27]	35	Photoplethysmography ECG	LSTM(SpO2)	92,9			95,5	AUC 0,98, PPV 99,2
			LSTM(IHR)	99,4			89,0	AUC 0,99, PPV 82,4
			LSTM(spo2+ IHR)	84,7			92,1	AUC 0,99, PPV 99,5
			LSTH(IHR	99,4(g?)			-	-
[43]	52	Airflow	CNN(DOSED)				0,81%	f1=0,57% +- 0,23
		PSG	**Expert Annotations**				**0,75%**	**F1=0,55% +-0,22**

Table 2.2: Table of papers



(a) IHR centroids



(b) SpO2 centroids

Figure 2.13: Centroids of IHR and SpO2 from vectors [27].

in Figure 2.13. The training of their model is done on a dataset consisting of one-minute segments containing apnea annotations for each minute of data. The one-minute segments are labeled as being either apnea or non-apnea minutes.

One of the more interesting measurements of accuracy is from Thorey et al.(2019) where they used five different technicians to annotate the same data. They then compared the results to each other using a majority ruling as a ground truth. They only reached an inter-scorer accuracy of 75%, which means that even sleep technicians annotations used for ground truth are not guaranteed to be correct.

There are many ways to detect apneas without doing a full polysomnography. Some models use data from sensors related to the heart and pulse, expanding and contracting body movements from breathing or oxygen levels in the blood. We see that these sensors can be applied in the machine learning models for predicting, and most are getting good results.

When comparing the different approaches, there is an important thing to consider: they are using different ways of reporting results. One paper uses only the AHI classification as ground truth. It predicts and compares only AHI scores of either non-apnea, which they define as AHI less than 15, or having apnea which they define as AHI of more than 15. This means that the ML model is a binary classifier. If this were to have any clinical significance, it should have more precision than just if the AHI is above or

below 15, and preferably it should predict when the apnea events happen.

When you compare an approach like this to an annotated polysomnography, you see a large difference between a polysomnography and machine learning predictions. The sleep technicians pinpoint their exact predictions for when the apnea starts and ends. They do this by analyzing all the different signals simultaneously to predict confidently. Doctors can not give a patient a diagnosis if they cannot explain why the diagnosis is predicted. In a analyzed polysomnography, they can look at all the sleeping data related to each other and pinpoint exactly why a classification was made. In many of these ML models, they only output a prediction of AHI values for the whole night or classify a time window as either normal or abnormal. There is no way for the doctor to see and understand why the prediction was made. It may help to show images of the recorded sleeping data for the predicted time, or an explainable framework for AI could be used so that the doctors can understand what is happening.

The advantages of these machine learning approaches are that they are quick to compute predictions. Even though none of the papers talked about the running time of the model, it is fair to assume it is done almost instantaneously. Training the model on the data is the slow part of using machine learning for prediction, but a new patient can be diagnosed quickly when that already is done. It is good practice to train the model on different people with different physiological features and different grades of sleep apnea. This is so that when a new patient comes in, the model hopefully already has trained for someone with similar features. As seen in the Table 2.2 the datasets are quite limited in size, and it may be hard to generalize for the population as a whole. As body types are quite dissimilar, a good dataset should balance humans with different features to avoid bias. However, as these are research papers, it is more important to show the possibilities of machine learning models rather than generalizing them for a final product.

Compared to a polysomnography, the sleep technicians time usage for annotation is magnitudes slower than a machine that can process the data nearly instantaneously. The time usage of a technician may only be limited by his employer regarding how much time can be used for one single patient, and a report from a polysomnography might take up to two weeks to deliver to the patient. Some of the patients might not even have apnea. This is probably the first thing an ML algorithm using fewer sensors can be applied for, to do a sort of pre-screening on the patient so that the polysomnography is mainly done on people with apnea. An important factor in this is limiting the number of false negatives the model predicts. A patient getting a negative prediction while actually being positive will be sent home without the needed treatment. This is not good for the patient, as all the negative effects of sleep apnea will still be there even though a breathing apparatus like a CPAP could significantly increase the patient's sleep quality. The advantage of a wearable sensor in contrast to a full PSG is that the test does not take up a hospital bed and does not need to be done only during a few or a single nights. A patient with a wearable sensor may use this sensor for a week, the first night may be discarded as people sleep

the first night differently with things attached to the body, and you can use the rest of the data to make a more generalized prediction than if you only had one single night. In that case, the ML frameworks with sensors win over the polysomnography as they can generate data over a much longer period than a polysomnography can. With multiple repetitions, it can become more certain.

Another comparison between the ML models and the polysomnography analysis is that a doctor may understand the patient better than an ML model. If a doctor sees patterns that he has never seen before, he can gather resources from his peers and earlier studies to understand what is happening and maybe even detect other diseases that are not apneas. A machine learning model will treat everybody the same based on the training data that has been fed to the network. A doctor may also see that a person has many events that almost are classified as hypopnea, but since they are not, they do not count towards the AHI index. A doctor may still classify this patient as an apnea patient and give the correct treatment because they see that they do not get the sleep they need.

Comparing experiments Looking at the implementation details from the papers referred to in Table 2.2 we see a large variance in the evaluation and parameters, making it difficult to compare them to each other. In these papers, they all report their findings using different parameters that make it hard to compare which models are working well and which are not. Some split the recording into time intervals of 10,15 and 30 seconds and classify each of these as either apnea or non-apnea and use this to generate performance results. Some use the AHI index calculated from these intervals to generate their performance results. The way we report these findings is important regarding how well we can understand the models. A good model should identify when each apnea event happens and have as few false negatives as possible. The datasets are heavily skewed towards non-apnea, making the accuracy metric give a high value that's not really fair because of the imbalance in the dataset. It is also hard to get a good accuracy as we can see an inter-technical accuracy of only 75%, which influences our training model. Maybe the models detect something on patient A because it has seen it on patient B, but the technician studying patient A did not classify this as an apnea. In contrast, if the technician that classified patient B had classified patient A, he would have classified this as an apnea. This may cause a lower grade of accuracy in the report. Therefore, it may be important that the predictions are understandable for the technician to judge if the algorithm is correct.

2.4.1 Sensors

The choice of sensors for detecting apnea using machine learning is important. It is directly related to which data we can analyze and can impact cost and sleep quality. As mentioned earlier, a polysomnography uses a large number of sensors. Still, a machine learning model may be able to see trends using fewer data points than a sleep technician needs,

and therefore we can use less invasive sleep recording techniques to only record the signals that it needs.

Single sensors. Many approaches use a single-sensor approach for training the machine learning model. One way to gather the data is using a single sensor from an available sleep dataset where sleep technicians have already annotated the apnea and hypopnea events. The advantage of this is that the datasets can be quite large, but the disadvantage is that there is no customization if you want to try something that is not recorded in the dataset. Another way is to gather a new dataset the normal way by doing a polysomnography and add an extra new sensor. Researchers can then have a sleep technician annotate the dataset using the polysomnography. The researchers can use this annotation as ground truth for training the model on the new sensor.

Multiple sensors. Multiple sensors are using more than a single sensor for predicting the apnea event. Since an apnea can have multiple causes, it may be relevant to include multiple sensors to increase the ML model's knowledge about the patient. An example may be the thoracic and abdominal expansion and contractions. When breathing normally, they are in phase, moving up and down simultaneously, and the body is therefore exhaling and inhaling since they are working together. If, however, they end up out of phase, there is no pressure change in the lung, and no air is inhaled or exhaled. If we were to just look at a single one of the sensors, say the abdominal sensor, we would just see a completely normal breathing pattern. However, by adding a thoracic sensor, we can see that the abdominal movement is not generating any airflow as the movements are 'countered' by the thoracic movements.

Chapter 3

Approach

In this chapter, we present an approach to (a) object detection models using Yolo to observe patterns and detect obstructive sleep apneas in images generated from abdominal breathing data, (b) create a tool, YoloApneaPredictor, around the learned model to analyze and evaluate a whole nights recording, and (c) build a client application, Yolo4Apnea, in cooperation with a server application for predicting obstructive sleep apnea in real-time.

3.1 Training the model

When we train a model for detecting obstructive sleep apneas using the Yolo object detection algorithm, we have some steps we need to follow. We will first be discussing how we preprocess the original dataset and we extract the relevant data. Secondly, we will discuss the conversion from raw data into plotted annotated images and the parameters that can be changed. Third, we will talk about the training of the model using these plotted images. An overview of the most relevant parts of this process can be seen in Figure 3.1

3.1.1 Preprocessing the data

In Section 2.1.4 we discussed the data provided in the sleep heart health study dataset. The dataset consists of 374 GB of data, including data from 19 sensors and annotations of all events occurring during a night. By removing parts that are not relevant for us, we can create a subset of the dataset that is more portable and manageable because of a smaller file size. The large dataset makes it difficult to work with when stored on the cloud as the latency is too high and the throughput is too low. Because of the coronavirus, a large portion of the research has to be done on a laptop with limited storage space. Earlier experiences have shown that external hard drives often disconnect, which is going to cause problems when training the model. Since there is not enough room for the dataset and all other required software and data, we need to compress the complete dataset as much as possible. We can extract only the needed features from the data

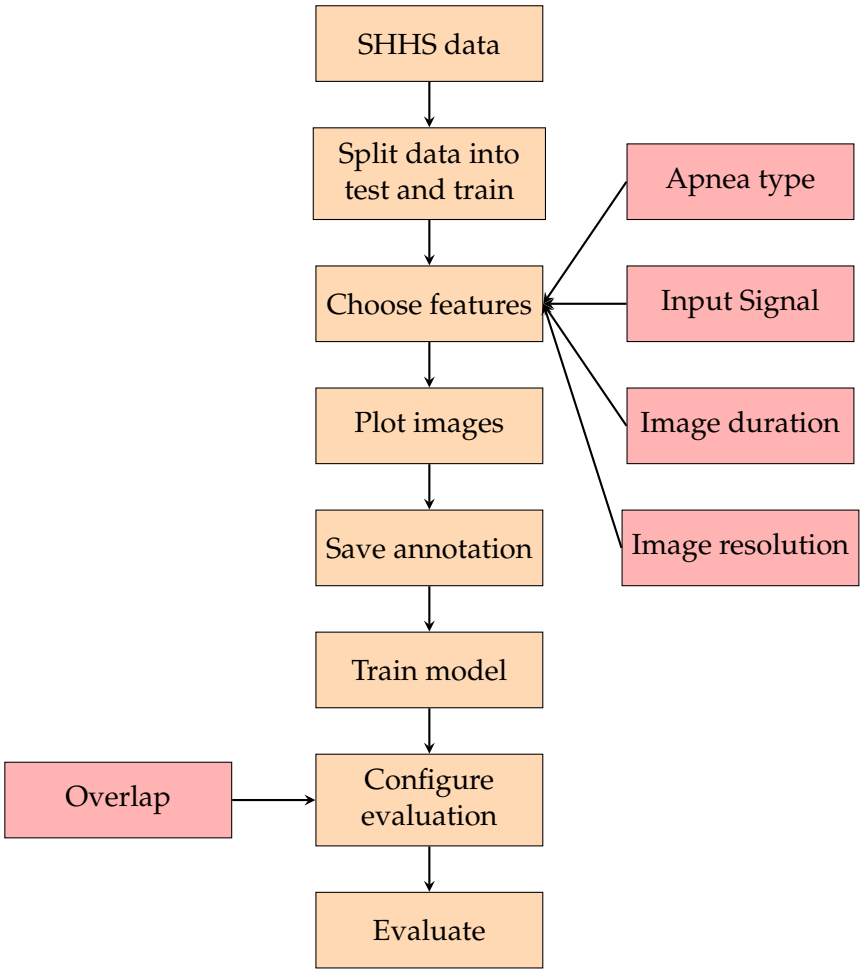


Figure 3.1: Overview of the process of training the ML model from raw data to evaluated model.

and discard the rest. By creating this subset, we can also read the data more efficiently using a more appropriate data structure for the data.

The choices we need to make when preprocessing this data are which signals we should keep, which annotation we should keep and the data structure for storing and pairing the signals and annotations. These choices influence the final data size used for training, but we limit the capabilities for training and evaluating our model later if we remove too much.

Choosing patients. We are lucky to have a large dataset consisting of data from 9884 nights of recording from 5804 distinct patients. Most of these recording having multiple apnea events scored, which gives us a large dataset to work on. We wish to have a model that generalizes as well as possible. Therefore we keep the data from all patients when preprocessing the data.

Choosing Signals. The Sleep Heart Health Study dataset contains recordings of 17 distinct signals in the first visit and 16 distinct signals from the second visit, as explained in Section 2.1.4. This thesis focuses on predicting obstructive sleep apnea from the expansion and contractions of the abdomen or thorax using a simple device to measure fluctuations in volume. We keep the signal from the abdominal plethysmograph and the signal from the thoracic plethysmograph. All other signals are then discarded as they are not needed for training our model.

Choosing annotations Every recording of a visit has an accompanying annotation file that a sleep technician has labelled. It is recorded in a .xml file with a data structure similar to Figure 2.5 from page 14. By traversing through the hierarchical data structure, we extract the events that have the labels: "Obstructive Sleep Apnea | obstructive sleep apnea", "Hypopnea | hypopnea", or "Central sleep apnea | central sleep apnea" as we wish to train a model for both obstructive sleep apnea and hypopnea events in addition to comparing how the model performs against a model trained on central sleep apnea. We store the events in a table with the start and end times in relation to the start of the recording. An example of the datastructure can be seen in Table 3.1

Data structure The data structure we choose after this preprocessing can be seen in Figure 3.2. It allows for small file size [23], a logical grouping of signals, data, and patients, and retains all the relevant original data from the Sleep heart health study dataset. Our data is stored as a collection of NumPy arrays of signals and annotations for each visit grouped together in files representing every patient.

We create a separate file for each patient when we store it on disk so that we can load each patient when needed. Every patient has partaken in the first study, but only a subset has partaken in the second study. We group both studies into the same patient's file, if applicable, or just the first if that is all that exists. The effect of storing only two signals, the NumPy file

	start	end	type
161	7865.3	7882.1	Obstructive
162	7898.0	7916.0	Hypopnea
163	7932.0	7958.0	Obstructive
164	7979.0	7999.0	Obstructive
165	8013.7	8042.5	Obstructive
166	8060.8	8080.1	Hypopnea
167	8087.0	8113.0	Hypopnea
168	8133.7	8148.9	Obstructive
169	8156.0	8172.0	Hypopnea

Table 3.1: Annotation table for a random patient showing scored events 161 to 169.

format, and the conversion from XML to tables reduced the file size from 374 GB to a much more manageable and portable size of 3,15 GB, which is only 0,84% of the original dataset, which is much more manageable and portable.

3.1.2 Plotting and annotating

The plotting of the images is vital for generating the training data for the training of the model. We need to convert the one-dimensional signal data to a graphical 2-dimensional representation in an image. MatPlotLib [22] is a plotting library with functions that allows us to plot one-dimensional array data as line graphs, but when plotting the images, we need to know what to plot and how large each image should be.

Annotations. We want to train our model on apneas, and since a full night recording contains a majority of normal breathing, we can choose to plot only the parts we know that contain apneas. We iterate through the annotations we saved in the previous section to plot one image for each apnea event. The position of the apnea is randomised in each picture to mimic the real-life scenario of an apnea happening at any time. If we were always to centre the apnea in an image, there is a possibility that the machine learning model would only detect apneas when they are in the middle of the image. When randomising the apneas position in the plot, there is also a large chance that part of the apnea is outside the image, and we only see a subsection of the whole apnea event. This trains the model to be more robust as we never know where in the picture the apnea is, and we also wish to detect the apnea as it occurs, not only when it is complete. Especially in a real-time setting, it is important that the model detects apnea when it has only seen a subsection of it.

Each plotted image is saved and annotated following the Yolo guidelines. The guidelines say that each image needs an accompanying text file that describes the class and location of each object in the image we wish the model to train on. The objects we are detecting are apnea events, and

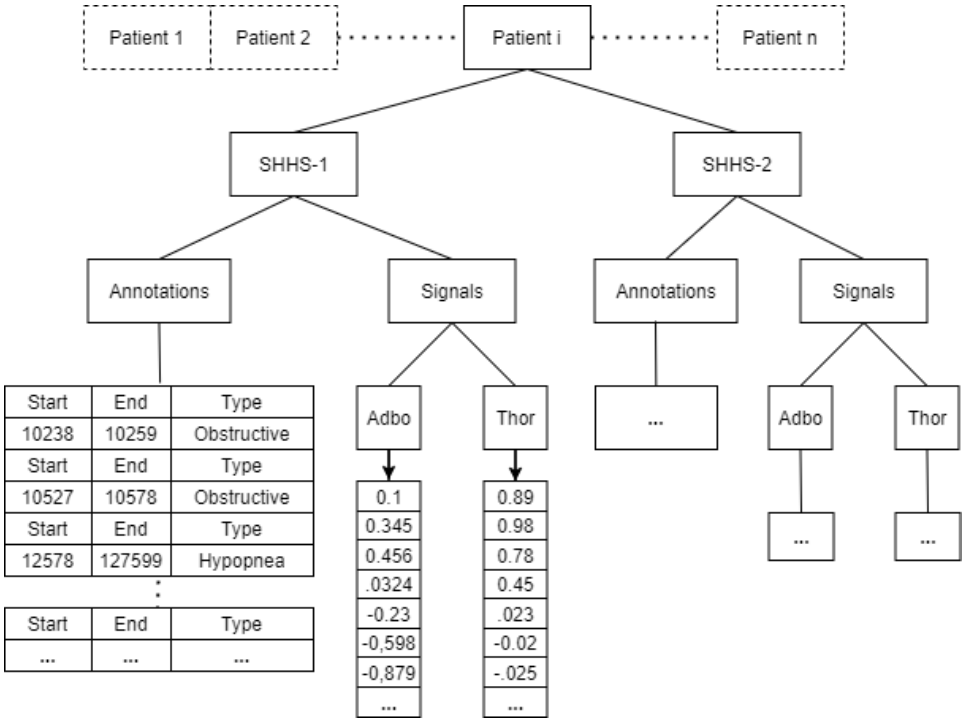


Figure 3.2: Figure of saved datastructure.

we label the class "apnea". Each label for an object in the image must follow the format of **"<object-class> <x> <y> <width> <height>"**. The x, y width and height values are percentage values relative to the width and height of the training image. Since we only have a single class we are detecting, the first value will always be 0, representing the class "apnea". The second value informs us of the centre of our apnea event in the x-axis of the image and will vary based on the position of the apnea in the image. We only care about the prediction boundaries on the x-axis as this is one-dimensional data projected as a two-dimensional image. Therefore, we wish to set the centre of the object to always be in the middle of the y axis of the image at 0.5, and the height value will always be one to fill the whole image. We, therefore, set the y value always to be 0.5 and the height equal to one. The width parameter will have to vary based on the duration of the apnea.

Image size The Yolo algorithm is robust regarding scaling images for predicting objects. When training the model, it resizes images automatically to be more robust, so the most important part for us is generating training images of high quality. We save our plotted images as .png images that are 1000px wide and 1000px high.

3.1.3 Training parameters

There are multiple parameters we can change when we are training the model. It is difficult to know which parameters affect the final result most, so the experiments need to be run with different configurations and be

compared later. Some of the things that can be changed are the selection of training and evaluation data, the signal type, the apnea types, and the display window.

Test-Train split It is recommended to split the dataset into three sets, as discussed in Section 2.2. We have two options for splitting the dataset. The first is splitting the dataset on apnea events, and the other is splitting the dataset on a patient basis. We choose to split the dataset on a patient basis for our approach. This means that all apneas for one patient are either included in the test set, the training set, or the holdout set. This allows us to evaluate our model on patients from the holdout dataset that we are sure that it has not been trained on, increasing the validity of our findings.

The first set we choose is the training set: This is the set that the model trains and learns on. We select the training set by choosing a selected number of patients and including all OSA events from this patient as part of the training set. The second dataset is the validation set. Its function is to prevent overfitting as the model trains on the test dataset. Finally is the holdout dataset, which is not used for training. In our case, the holdout dataset will be the remaining patients that are not selected in the first two datasets. The function of a holdout dataset is being a dataset that is kept separate from all other training. We will be using the holdout dataset as the dataset that we run our evaluation on.

Signals. There are multiple signals in the sleep heart health study dataset, and we have already chosen to focus on the abdominal and thoracic signals. We can choose to train on these signals, or we can combine them in some way. If we combine them, we can average the signal data, so we still have a single line to plot. An example of the two main signal types and how they are combined can be seen in Figure 3.3

Apnea Types In the dataset, there are 3 apnea types. Obstructive sleep apnea, hypopnea, and central sleep apnea. Variations of which of these chosen for our training and evaluation data can influence the quality of the model.

Display window. The apneas recorded in the SHHS dataset can last between 0.8 seconds and 256 seconds, with the middle 50% of values are apneas lasting between 17 seconds and 31.4 seconds. The number of seconds we include in our plot can influence how the model detects sleep apnea. A shorter display window for each image might lose some of the needed data before the apnea, which might be important for apnea detection. A long window might remove resolution in the data, making detection difficult for the model.

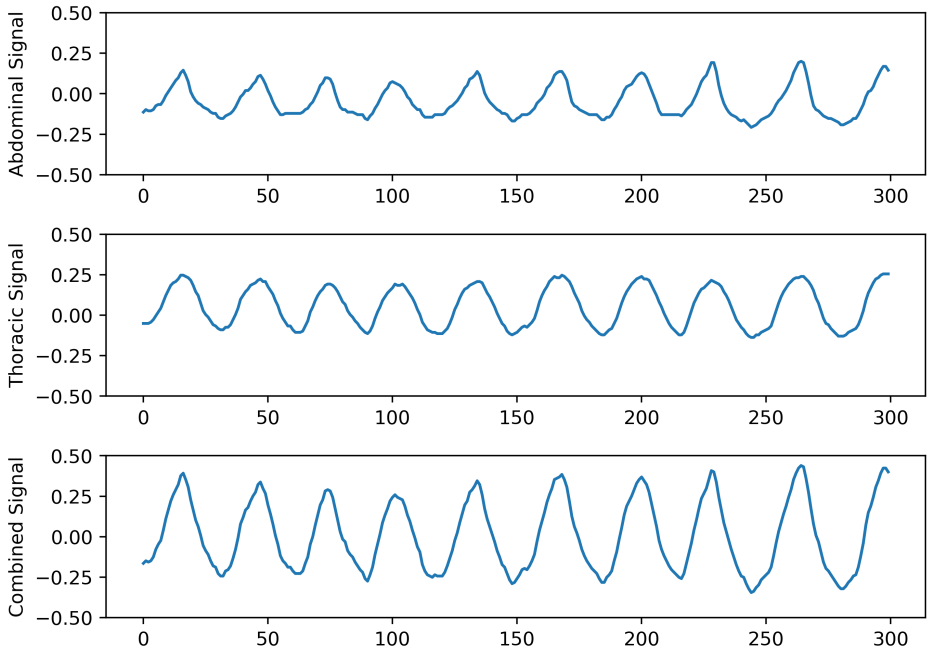


Figure 3.3: Plots of abdominal signal, thoracic signal, and the combined signal of abominal and thoracic signal.

3.1.4 Example of training data

An example of the plotting and annotation step results can be seen in Figure 3.4. In the image, we have plotted the abdominal signal on a 1000x1000 pixel image and labelled it according to our guideline. Here is an explanation of the labels:

0 *Object class:* The 0 represents apnea class

0.58 *Center of the object on X-axis:* The centre of the apnea is 58% into the image on the X-axis

0.5 *Center of the object on the Y-axis:* the centre of the apnea is in the middle of the image on the Y-axis

0.3033... *Relative size of the object in the image on X-axis:* The apnea last for 0.3033333...% percentage of the image

1 *Relative size of the object in the image on the Y-axis:* The apnea fills 100% of the y-axis of the image

When we use Yolo to train a model using all these parameters, we output a .weights file containing the trained weights for all the layers. In combination with a .cfg file with the layers and their configuration, we now have a trained network. These two files represent our trained model and can be used for inference.

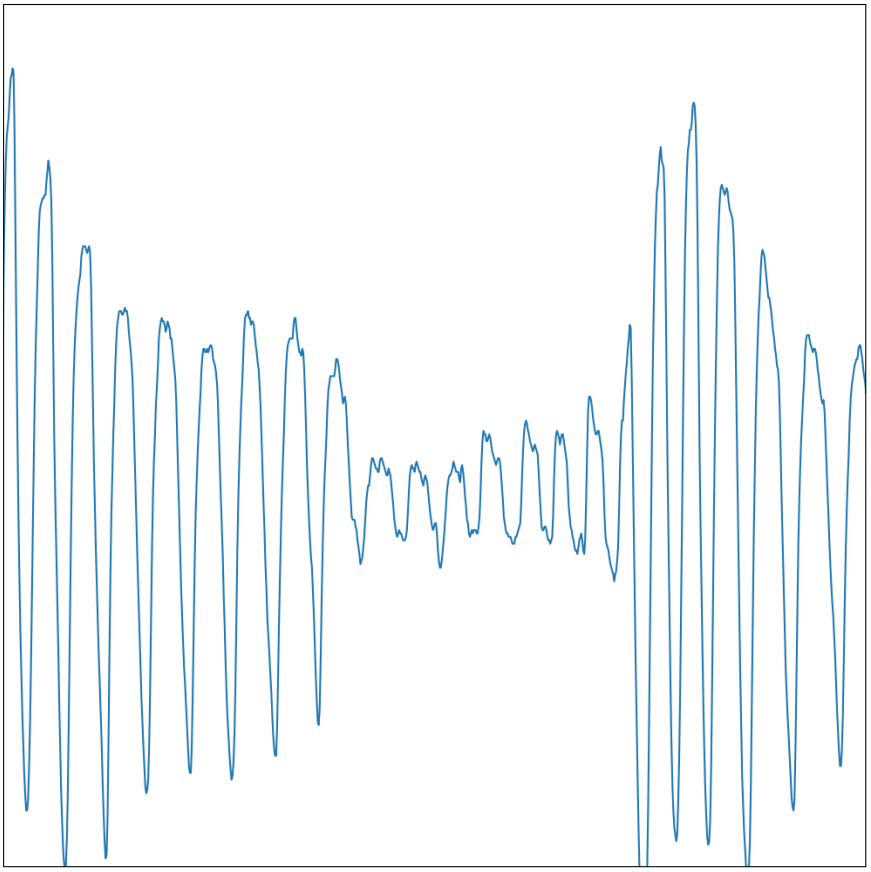


Figure 3.4: A plot with an apnea occurring in the middle of the image with label: **"0 0.58 0.5 0.30333333333333334 1"**

3.2 YoloApneaPredictor

In the previous section, we explained the process and the parameters we can change when training our obstructive sleep apnea detector using Yolo. This detector can now detect obstructive sleep apnea objects in single images. We expand this model to a detector capable of predicting multiple OSA events in a whole night sleep by developing a tool called YoloApneaPredictor. This tool wraps a trained model into a tool that generates images of the whole recording, including overlapping images, runs detections on these images, handles the predictions, and allows for comparing the predictions to the annotated files to measure the quality of the model which we will be doing in Section 4

Interface

We create YoloApneaPrector as a tool that can handle many different evaluation settings. It ensures that we evaluate our models in comparable settings, with only the parameters changing. It can be initialized with parameters affecting the detection of the apneas. These parameters are:

1. ML model
2. Types of apneas to look for
3. Duration of each image generated
4. Overlap between each image
5. Size of images generated
6. Minimum confidence threshold
7. Non-maximum suppression threshold

The YoloApneaPredictor has two main functions. The first is inserting signals to predict, and the other is getting predictions and comparisons to the annotated ground truth. The append signal function receives a new signal and runs detection on the data. This function abstracts the image generation and handles creating images and predicting apnea events. This function can be used in the real-time model as it handles the concatenation of the new data with the previously received data and handles the image generation of the correct new parts.

Loading the model

The YoloApneaPredictor can be configured with any Yolo model. We have chosen to use OpenCV's DNN module [25] for loading the trained model as it can be used to run inference on our model from python. The flexibility of being able to load any model allows us to compare and evaluate multiple models to see which performs best in a robust manner.

Image generation

The images we plot from the signal data use the same plotting script we used to train the model. This helps improve validity as the only difference between the images we run inference on and the images we have trained on is the signal data. We can either generate images of distinct signal data segments, or we can create images that overlap each other somewhat. Overlapping means that we predict the same signal data multiple times. We do this as our model might be better at predicting apneas if it sees the abdominal and thoracic movement either before or after the apnea. Figure 3.5 shows how we can generate images with overlap. In this figure we use a signal duration of 90 seconds for each image, and an overlap stride of 45 seconds between each image generated.

Predicting apneas

When we run an inference on a generated image, the model returns the x and y positions of the lower-left corner and the height and width of all objects it detects in pixel values. Since we represent a time series as a 2d image with time on the x-axis, we define that a detection of an event covers

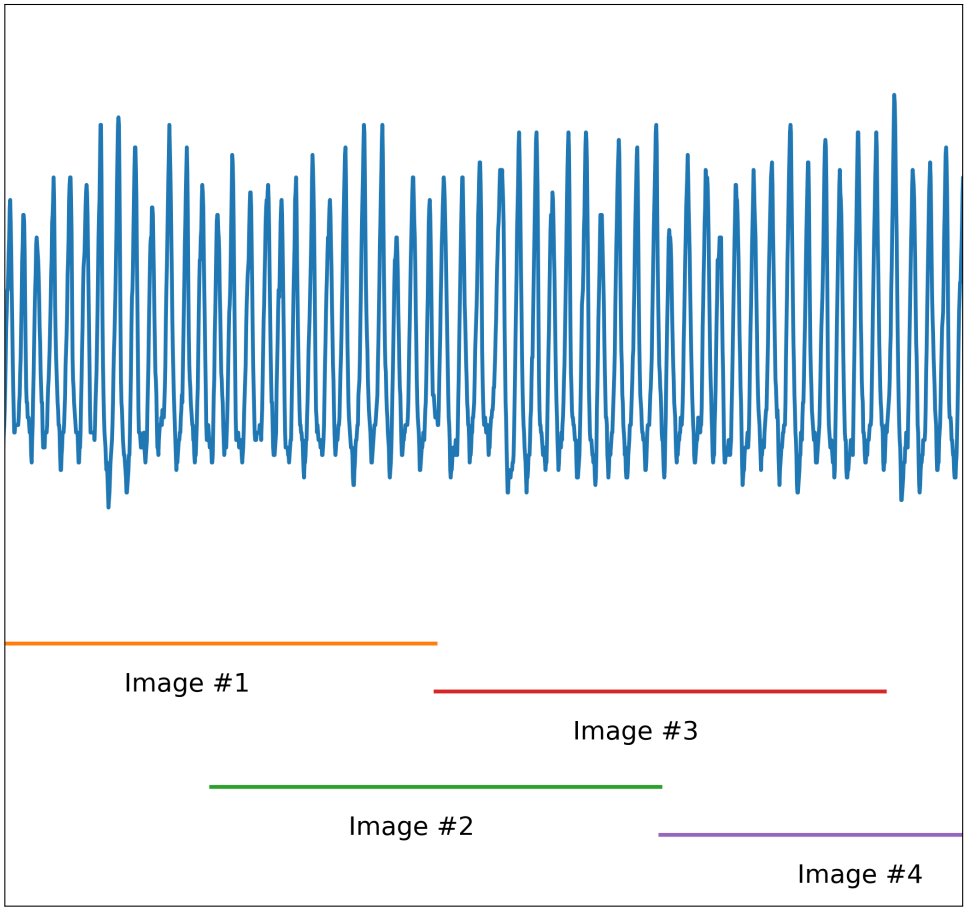


Figure 3.5: Example of how multiple images overlap the same signal data. Example is using a window size of 90 seconds and overlap stride of 45 seconds.

the whole image in the y axis. Therefore we only retain the x position and width.

As our final representation of the predictions of obstructive sleep apnea during sleep is represented as a floating-point array with indices for every decisecond, we need to convert these relative positions to a fixed position with an offset based on image size in the array. The method for inserting the prediction can be seen in Equation 3.1, 3.1 and 3.3where P_n is the prediction array.

$$INDEX_{start} = Image\_Start + X_{pos} * Window\_size \tag{3.1}$$

$$INDEX_{end} = Image\_Start + (X_{pos} + width) * Window\_size \tag{3.2}$$

$$\forall n \in \{INDEX_{start}, ..., INDEX_{end}\}.P_n =_{max} (P_n, New\_prediction_n) \tag{3.3}$$

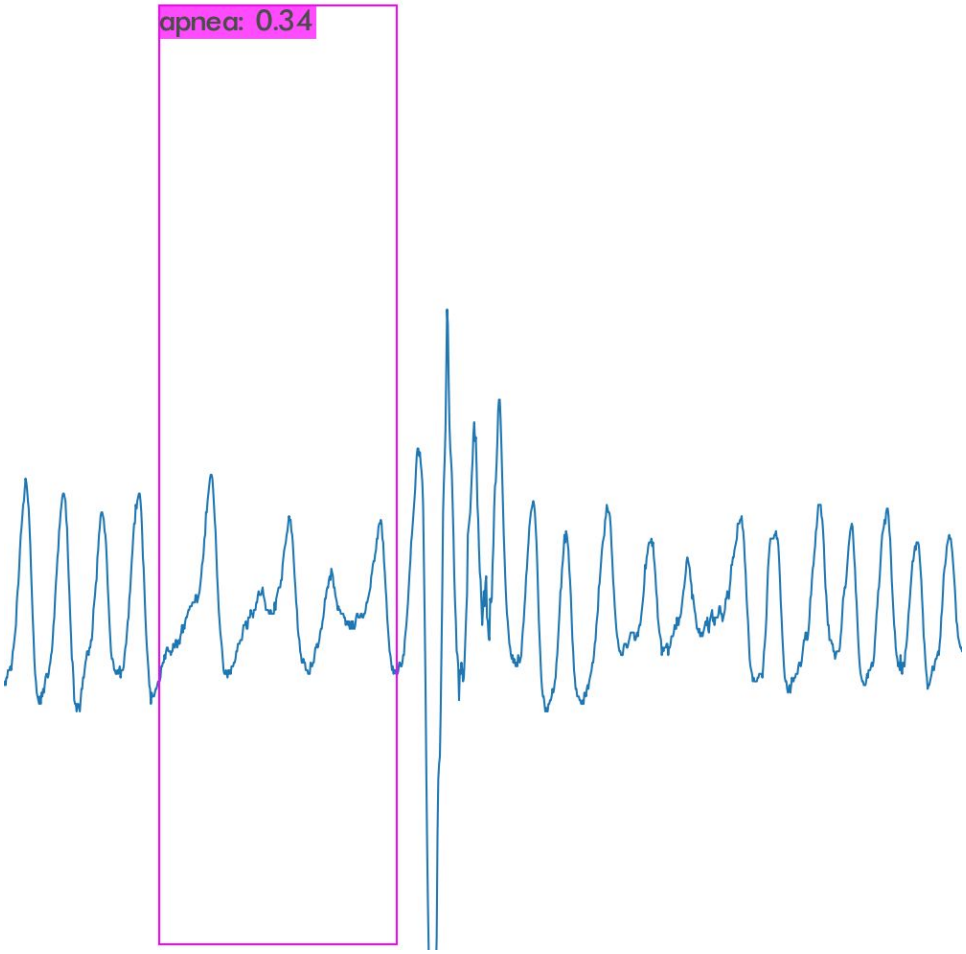


Figure 3.6: Figure of Yolo bounding box on generated image.

As there will be multiple images overlapping the same region of time from the signal data, this algorithm will always retain the prediction with the most confidence, even when multiple detections are occurring for the same time. See also Figure 3.7, which shows an abstraction of how an apnea is detected in signal data and inserted into the previously detected apneas, still retaining the prediction with the highest confidence.

Ground truth

The ground truth data we stored in section 3.1.1 was stored as tables with the type of data as one column, the start time of each apnea in another, and the end time as another. To evaluate our predictions effectively, we can convert these annotations to the same one-dimensional-array data structure as we use for storing our predictions. The annotation array starts by being represented by a zeroed array of the same length as the signals from a patient in deciseconds. Each index in the array will then be representative of the time in deciseconds since the recording start. We then iterate over every apnea event in the table and fill the array's values

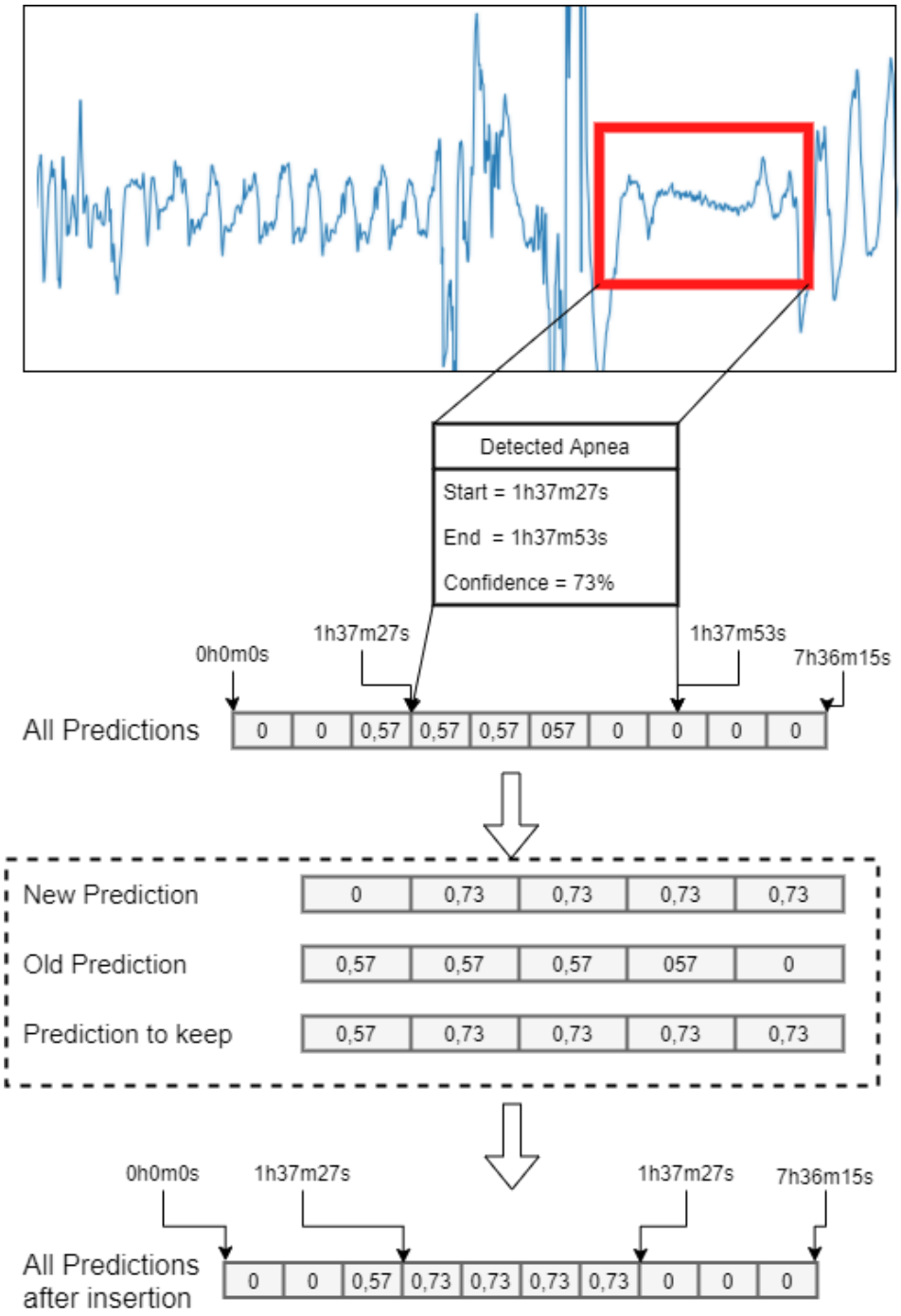


Figure 3.7: The process from detecting an apnea and inserting it into the whole prediction array

from the start index to the end index with a value representing the type of apnea in the annotation. If there is no apnea event registered, the value will be 0. If there is obstructive sleep apnea, the value will be 1, and if it is a hypopnea, the value will be 2.

Comparison

As we now represent the prediction and the ground truth as arrays, we can easily extract scores. We pad the ground truth array to the same length as the prediction array with zeroes to have the same size. The ground truth array may be shorter than the predicted array because the ground truth array is only of the length of the last observed apnea, while the prediction array is observing the complete data. The reason for padding the data with zeroes is that there is no apnea occurring here in the annotation data. Since we now have two arrays of the same length, we can easily use the scoring functions from Scikit-learn [1] which ensures that we score our model using functions that have been rigorously tested. The scores we use are detailed in Section 2.3 and include scores like accuracy, MCC, f1, precision, and AUC.

3.2.1 Yolo4Apnea

Yolo4Apnea is the real-time tool developed to see the detection happening in near real-time. An example of the interface can be seen in Figure 3.8. It consists of a server connected to the YoloApneaDetector backend and a client that interface with the server that is running the machine learning model. The client visualizes all apneas in a more user-friendly way.

Client

Yolo4Apnea client is a real-time tool connected to the Yolo4Apnea server. It is made to be a visualization of the predictions from YoloDetector. It features a scrollable graph of the signal with predictions including confidence superimposed onto the graph, a list of all predictions, and metrics like the AHI index for the signal. It sends the most recent signals to the Yolo4Apnea server and receives predictions from the latest 90 seconds when running. It always overwrites the last 90 seconds of predictions with the new information received, as the newest received information should always be the most correct information.

The web interface also allows viewing specific apneas and seeing the calculated metrics of the recording. It can either be used in the real-time mode where each signal is analyzed when the signal is generated, or it can be used for predicting a stored signal to get measurements from a whole nights recording.

Server

The Yolo4Apnea server provides an API endpoint to the Yolo4Apnea Client using the Python library Flask. It receives new signals from a client and passes them to the YoloDetector interface explained in Section 3.2. The API endpoint will then return the latest 90 seconds of predictions to the client. The reason for returning the last 90 seconds is that YoloApneaPredictor can detect new apneas anywhere in the last viewing window. By returning the

last 90 seconds, we make sure that we always view the predictions that the model has been most confident in during this viewing window.

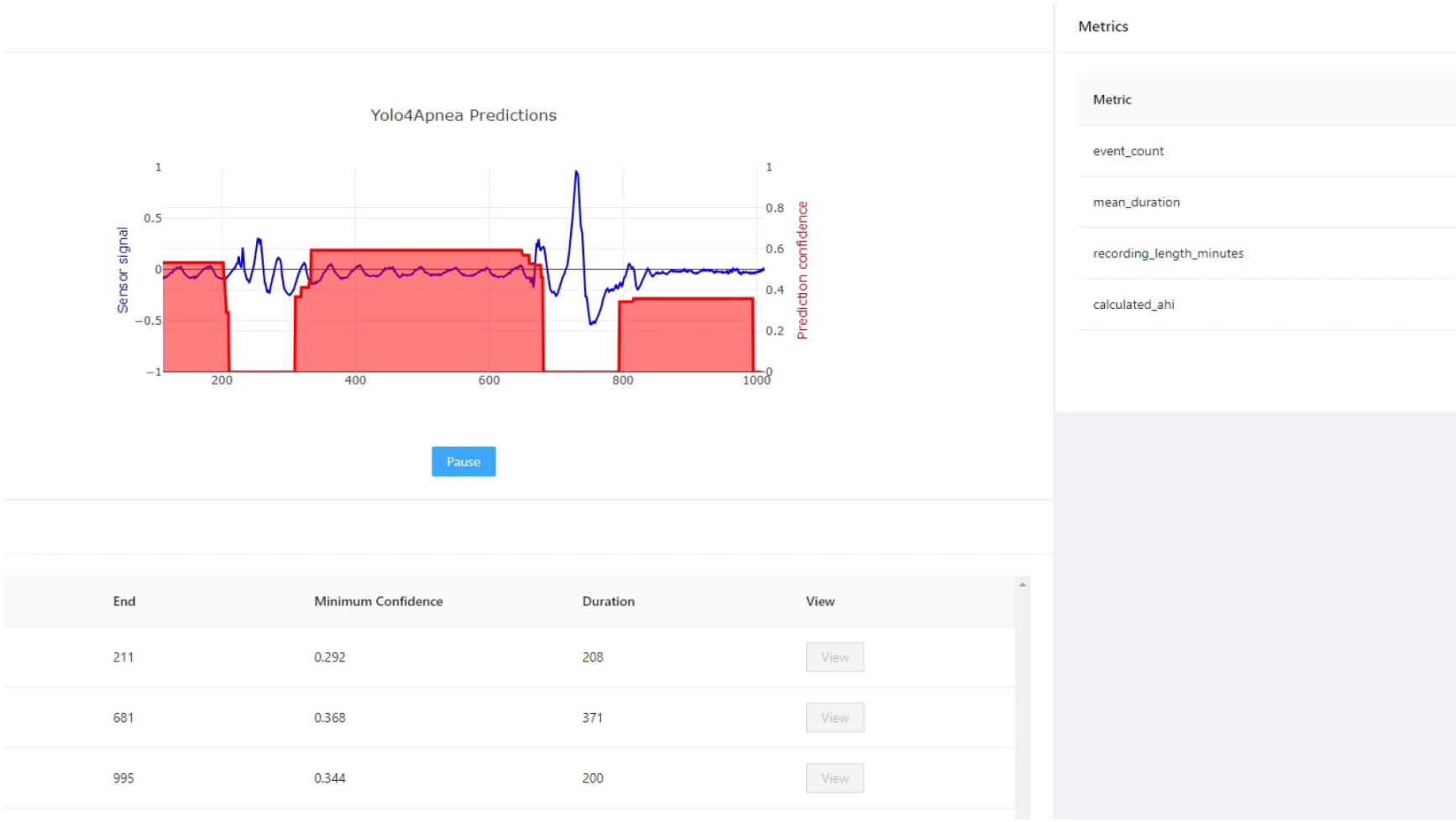


Figure 3.8: Screenshot from Yolo4Apnea web interface with multiple apneas.

Chapter 4

Evaluation

4.1 Research questions

In Chapter 1 we proposed three research questions that we wish to answer in this thesis. In this section, we will explore the reasons behind selecting the chosen research questions, and we attempt to answer the questions.

RQ1: How does changing hyperparameters influence the performance of an object detection model for detecting OSA?

We are doing a novel approach for event detection in a time series where we are rendering the one-dimensional array as a two-dimensional image and using an object detection algorithm on it to detect events. Since there is no heuristics for this, we need to explore multiple settings to figure out what performs well and what doesn't. When attempting to answer this question, we will attempt to answer how the hyperparameters we set and the features we select from the data influence our model's evaluation. We have chosen this question as the first question to answer as it influences the next research questions greatly and could be an important part of detecting sleep apnea.

The guidelines for Yolo inform us that it is recommended to train for a minimum of 6000 batches, or 2000 batches times the number of classes we wish to predict if the resulting number is higher than 6000 [2]. To ensure that the model receives enough training time, we have chosen to train each model for 10000 batches. Experimenting shows that training using the Yolo machine learning algorithm for 10000 batches takes around 9 hours using a Nvidia RTX3080 graphics processing unit. In addition to the training, there is also the time needed to generate the training images and the time needed to evaluate the trained model. This cost in training the model makes it expensive to check many configurations of hyperparameters. Therefore, we need to be selective regarding which configurations we wish to evaluate.

Hyperparameters are parameters used for tuning the machine learning model's performance. It is impossible to know beforehand what the optimal parameters are, so we need to manually or automatically search

some possible solutions and choose the parameters where the learned model performs best according to some predefined metrics [10]. We will focus on changing one parameter at a time and comparing the results to each other, retaining the parameter value that performs best. This type of gradient descent has a large chance of ending in a local minima instead of the global minima but is an effective way to improve our model when the search space is large and expensive to search.

In Section 3.1 we explored some of the parameters that can be changed in our model. In this section we will evaluate the effects on our model by changing these parameters:

- Signal type
- Window size
- Window overlap
- Sample size
- Apnea Types

We will be evaluating the model based on the MCC score and the f1-score as mentioned in Section 2.3 because they are good metrics for comparing imbalanced dataset. We will mainly focus on maximising the MCC score and keeping an eye on the f1-score and the AUC score. The MCC value is a balanced measure that is no better than random chance when the value is 0 and is a perfect predictor when the value is 1. If a model has quite a similar MCC score but improves significantly in one of the other scores, we will keep that value as the best result. The model with the best result will be the model that is used when answering the next two research questions.

Results

The following results are all trained using the same pipeline to ensure comparable results. All random selections are equal for models with the same number of training patients or evaluation patients. Each model with the same evaluation size and training population size has the same patients used for training, testing, and evaluation. This also ensures that every model we are comparing in each section is evaluated on the same data as the others, ensuring comparable results. Every model is trained on a dataset made from data from 100 patients and tested on data from 40 patients. The signal type, the window size, and the overlap models are evaluated on data from 20 patients. The sample size models are evaluated on data from 20, 100, and 500 patients, and the apnea types models are evaluated on 100 patients. A summary of all results and the parameters can be found in Table A.1 in Appendix A. We have divided the research question into five sub-questions:

RQ1.1: How does the chosen signal affect the performance of our model? In this thesis, the focus is on using non-invasive and easy to use sensors for detecting sleep apnea instead of doing a polysomnography. We wish to use plethysmograph sensors that measure volume changes and have chosen the thorax and abdomen sensors for this task. These two sensors give us two distinct signals to select from. One is the signal from the abdominal sensor, while the other is the signal from the thoracic sensor. We can also combine these as one signal. Table 4.1 shows the comparison between training a model using the abdominal signal, the thoracic signal, and the combined signal of both signals.

Signal type	F1	MCC	AUC
Abdominal	0.568	0.428	0.777
Combined	0.571	0.455	0.819
Thoracic	0.520	0.387	0.752

Table 4.1: Metrics for different signal types.

Table 4.1 shows the f-1 score, the MCC score, and the AUC for each of the three signal types. The MCC score we have chosen to focus on is the highest with a value of 0.455 when using the signal consisting of a combination of the abdominal and thoracic signals. The highest f1-score is also observed when using the combined signals. Similarly is the AUC also the highest score when the model is trained on the combination of the signals. A model trained on the signal from the abdominal sensor appears to be the best single sensor model. The thorax sensor scores the lowest on all relevant parameters. We continue to use the combined signals from the two sensors for our hyperparameter exploration as these have the best results with all other parameters equal.

RQ1.2: How does the chosen window size affect the performance of our model? Window size represents how many seconds of signal data each image used for training the model contains. A higher value will keep more data about what happens before and after an apnea but will have less detail as the signal is more compressed. It also has a higher probability of containing the whole apnea, not just parts of it, which might affect the performance. Table 4.2 compares the evaluation metrics with sliding window length values in the range of 30 seconds - 180 seconds with 30-second intervals.

In this table, we have 6 distinct experiments. The highest MCC score occurs when the sliding window is 120 seconds with a score of 0.456, closely followed by the MCC score when the sliding window is 90 seconds, which is 0.455. We see lower scores when we increase the sliding window and when we decrease it. The sliding window of 120 seconds also has the highest score on the f1 metric, but the AUC is quite close to the scores from the 60 second and 90-second models. We choose to keep the 120-second sliding window parameter value for our next steps in experimenting. One

Sliding window (s)	F1	MCC	AUC
30	0.413	0.274	0.662
60	0.541	0.421	0.815
90	0.571	0.455	0.819
120	0.576	0.456	0.780
150	0.528	0.423	0.695
180	0.457	0.370	0.650

Table 4.2: Metrics for different window sizes.

thing to remember when viewing these metrics from this search is that all other parameters are kept the same, including the window overlap. There is a high chance that the window overlap is highly linked to the sliding window duration, but evaluating all the possible combinations is too computational difficult within the scope of this thesis. This means that there is a high probability that a more exhaustive search in combination with changing the window overlap value may return other promising configurations as the window size and overlap probably are highly linked.

RQ1.3: How does the chosen window overlap affect the performance of our model? The window overlap is the stride we move forward between each picture we generate when evaluating our model. A lower overlap value will mean that we predict apneas in the same period multiple times, while a higher overlap stride value will predict fewer times in the same period. In a real-time version, we should optimally have a low value to predict as often as possible. Table 4.3 shows the results of a model trained on the combined signal, with a 120-second sliding window with overlap strides of 15 seconds, 30 seconds, 45 seconds, and 60 seconds.

Overlap stride (s)	F1	MCC	AUC
15	0.560	0.445	0.819
30	0.575	0.456	0.800
45	0.576	0.456	0.780
60	0.577	0.455	0.770

Table 4.3: Metrics for different window overlap values.

The highest MCC score occurred when the overlap is 30 seconds with a value of 0.456. The highest f1-score was when the overlap was 60 seconds. The AUC scores increased when lowering the overlap values, ranging from 0.770 to 0.819. When comparing the consequence of changing this parameter compared to the previous parameters, we see that the effect is diminishing. The lowest MCC score in this pass was 0.445 and the highest was 0.456. In Table 4.1 the range of MCC was between 0.387 and 0.455.

RQ1.4: How does the chosen sample size affect the performance of our model? Predicting obstructive sleep apnea and evaluating the results

using our model usually takes between 1-2 minutes for each recording. Each patient has a minimum of 1 recording and a maximum of two. The signal type, sliding window size, and overlap evaluation were evaluated using a sample size of 20 patients. Because the variation in scores when changing parameters diminished greatly, we wanted to see the effect of evaluating our model on varying sample sizes. We assume a large variation between patients and therefore evaluate three models with varying sample sizes to see how this affects the results. These sample size values are 20, 100 and 500. These experiments were evaluated using the same parameters as previous experiments, but with a sliding window overlap of 45 seconds. Table 4.4 shows the results from these experiments

Samples	F1	MCC	AUC
20	0.576	0.456	0.780
100	0.570	0.448	0.772
500	0.548	0.432	0.765

Table 4.4: Metrics for different sample sizes.

We can see a noticeable deviation between the metrics when changing the sample size parameter. The f1-score has a standard deviation of 0.015, the MCC has a standard deviation of 0.012, and the AUC has a standard deviation of 0.007. A higher sample size will give us more accurate results, but evaluating our early models on 20 patients is a good tradeoff between accuracy and evaluation expensiveness. The gain from the tuning of parameters is more than the standard deviation. The score was lower for each metric when we increased the sample size, which might be because the model evaluates a larger variation of people, many of who might have physiological features that the model has not trained enough on yet.

RQ1.5: How does the chosen apnea types affect the performance of our model? This focus is on detecting obstructive sleep apnea, but as mentioned in Section 2.1 there is another type of sleep apnea called central sleep apnea. We can try our model on this type of apnea to see how well it performs. It should be made clear that similarly to obstructive sleep apnea, we also use the hypopnea events for training and evaluating our model. There is no distinction in the annotations by sleep technicians, or in the Sleep Heart Health Study dataset, about whether the hypopnea is caused by an obstruction in the airway or a central blocking. In Table 4.5 we see the results from training and evaluating a model for central or obstructive sleep apnea events together with hypopnea events using the previous parameters and a stride of 60 seconds.

This table shows us that we get the best result when we have a model trained for obstructive apnea. Why this worked better could be explored further, but it might be because there are a lot more obstructive sleep apnea events than central sleep apnea events in the dataset. Another possibility is that during an obstructive sleep apnea event, the body maintains breathing

Apnea Types	F1	MCC	AUC
Central and Hypopnea	0.499	0.401	0.740
Obstructive and Hypopnea	0.570	0.450	0.766

Table 4.5: Metrics for different apnea types.

attempts even though the airways are blocked. An obstructive sleep apnea event occurs when the brain is not sending signals to the body to attempt breathing, which might not be discoverable from the polysomnography signals as well as the obstructive events. Another possibility is that we have optimized our parameters for obstructive sleep apnea events and are now applying the same parameters to central apneas without doing the same hyperparameter search. Therefore, it is not a good comparison, but it is a topic that could be investigated further.

Summary and discussion

In Table 4.6 we see the parameters that gave is the best possible result.

Signal Type	**Sliding Window**	**Overlap**	**Apnea Types**
Combined	120 Seconds	30 Seconds	Obstructive and Hypopnea

Table 4.6: Parameters chosen for best results.

F1	**AUC**	**MCC**	**Accuracy**	**Precision**	**Recall**
0.549	0.784	0.432	0.828	0.445	0.667

Table 4.7: Averaged scores from 500 samples.

In Table 4.7 we can see the average scores from 500 sampled patients. We are in this evaluation comparing the predictions to the ground truth in a decisecond interval. This means that if we are not exactly predicting the start and end of the sleep apnea event, we will get lower scores, even though a few seconds earlier or later might not matter too much. The sleep technician annotating the data is not this exact. The accuracy score we end up with is 82.8%, but this metric does not tell us too much because of the imbalanced dataset. The MCC-score of 0.432 is a good indicator that our model is better than random chance and promising for the future of the model. Our precision is 0.445, and our recall is 0.667, which indicates that our model is far from perfect yet. It should also be noted that this is a novel way of detecting obstructive sleep apnea, and there is a large chance that this model can be improved further.

In Section 2.4 we explored a paper by Thorey et al. who had looked at the accuracy of five sleep technicians annotating the data and reaching an inter-scorer accuracy of 75%. Our model is trained on a dataset from many sleep technicians from multiple hospitals over a large period of time. It isn't easy to know if our prediction of a sleep apnea event could have been a true apnea event if another technician annotated the data.

RQ2. How well does Yolo work for detecting OSA compared to human annotations?

Introduction

When we attempted to answer the first research question, we searched for good parameters for a machine learning model focusing on good f1-scores and MCC scores. We found the best performance with a model trained on a combination of abdominal and thorax sensors, a sliding window size of 120 seconds, an overlap of 30 seconds between each image, and looking for both obstructive apneas and hypopneas. That model was evaluated using single metrics like MCC and f1-scores representing the performance. When answering this research question, we wish to answer how well this model performs from a clinical perspective.

We are using the same model with the same training parameters except for the sliding window duration. We are evaluating two versions of this model. One where the sliding window value remains at 30 seconds, this model is used for analyzing and predicting the severity of obstructive sleep apnea for whole night sleep. We evaluate the other model where the sliding window value is one second. This one-second overlap model is meant to represent a real-time detector of sleep apnea where it is more important to discover all events quickly after they happen. We will present these two evaluations after each other and see how well they detect apnea events.

30-second overlap evaluation

In this section, we have evaluated our 30-second overlap model on 500 patients consisting of one or two recordings each. We have explored the quality of our model by exploring more evaluation metrics than the previous sections. These are the ratio between the recall and specificity at different confidence thresholds, the calculation of the predicted AHI value and comparing it to the human annotation AHI values, and looking at a confusion matrix using apnea events instead of decisecond precision.

ROC. The Receiver operating characteristics curve shows the performance of a binary classifier at different threshold levels. Figure 4.1 shows the ROC curve for our model. The red dotted line shows the performance of a completely random binary classifier, while the blue lines show the curves for all our recordings. The thickest blue line close to the centre shows the average ROC performance of our model, while the slightly coloured blue area shows the boundaries of one standard deviation of the average ROC curve. In our case, we can see that our classifier is performing better than random guessing but still having a large variance between patients.

Confusion matrix. The confusion matrix is another way to present prediction quality. In this matrix, we have defined a true positive apnea event to be an event where we have detected more than the whole event, the whole event, or a subset of the event. A false positive event is an

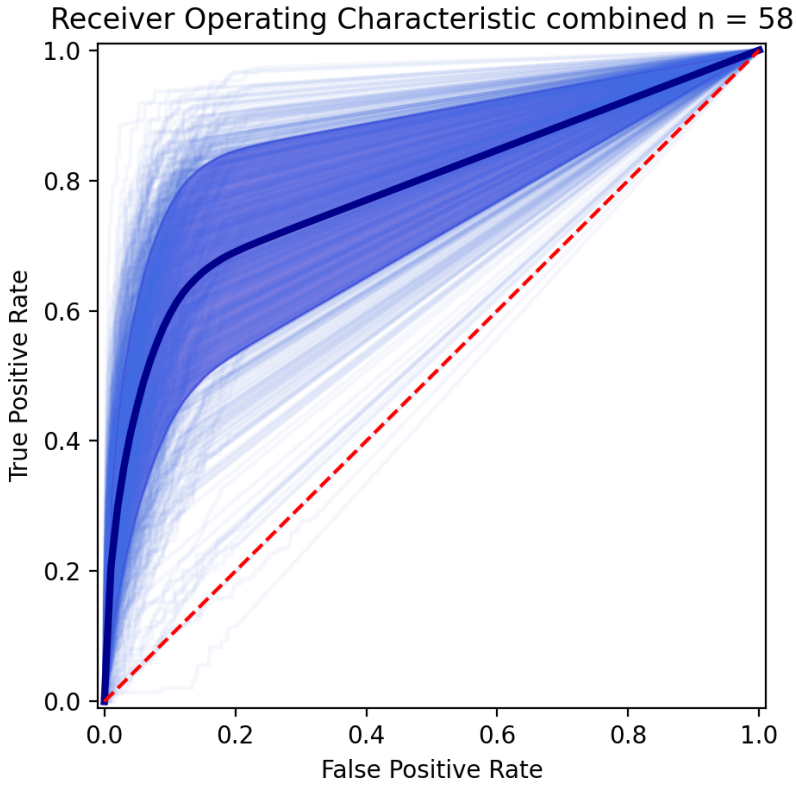


Figure 4.1: ROC curve of Yolo model.

event we have predicted which not appear in the annotated file. A false negative event was an apnea event where there was an apnea, but we did not detect an apnea at all. The reason for evaluating the model like this is the dependency on counting events when calculating AHI. A metric like this focuses on which apnea events we detected instead of the exactness of the start and end of the apnea. Table 4.8 shows the confusion matrix for events.

Type	**True Positive**	**False Positive**	**False Negative**
Apnea count	96693	71154	42307
% of all apneas	69.56%	51.19%	30.44%

Table 4.8: Confusion matrix.

Of the 500 patients evaluated, there was a total of 139000 apnea events occurring. Our model predicted 96693 of them, or nearly 70% of these correctly, and missing around 30% of apneas. In addition, we predicted 71154 apneas that were not annotated in the ground truth data. This number is very high, and compared to the true positive events, it is around a 51% chance of a detection being a false positive apnea. These results show many apneas going undetected and many false predictions.

AHI prediction vs ground truth. The AHI index indicates how severe a patient sleep apnea diagnosis is (see Section 2.1.2). The AHI number is calculated as the number of apnea and hypopnea events occurring per hour of sleep, with higher values being more severe than lower. Figure 4.2 is a scatterplot of our predicted AHI values against the calculated values from the ground truth.

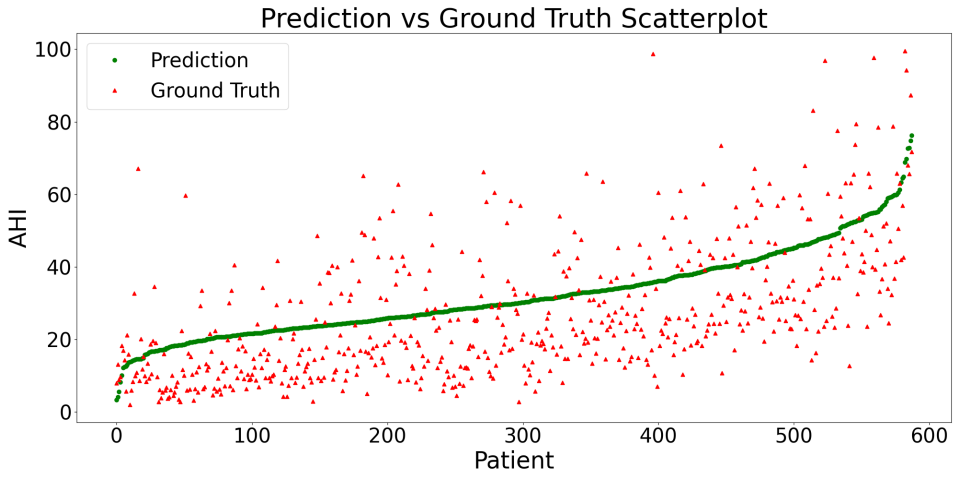


Figure 4.2: Predicted AHI index vs true AHI index.

We calculate our predicted AHI value by counting distinct segments of predicted apneas and divide by the hours of recording in the signal file. We calculate the AHI value of the ground truth as the number of annotations divided by the hours of recording in the signal file. The plot in Figure 4.2 is a plot of our predicted AHI index represented by a green dot against the true AHI represented by a red triangle. It is sorted in ascending order based on the predicted AHI index. A visual inspection of this plot shows a large variance in the calculated value, where we generally predict a higher AHI index then the AHI index that was calculated from the dataset. A theory of why this might happen is that our model are predicting more apneas in each dataset because it has trained using other sleep technicians annotation that might have scored the apnea as an event. The labels we use for our ground truth are human annotations that have been shown to differ when different sleep technicians score the same recording.

One-second overlap evaluation

We are also looking into the real-time capabilities of our model. We got the best results from our model when analyzing patients with a stride of 30 seconds, but this stride is too coarse to detect apnea in close to real-time. We, therefore, wish to try evaluating the model using a stride of low value; in our case, we test it with a stride of 1 second.

For an 8 hour recording with a stride of 30 seconds and 120 seconds sliding window, we need to generate 957 images of the breathing pattern that can be fed into the Yolo network. When evaluating our model in real-

time and changing the stride to 1 second, we have to generate 28681 images for each 8-hour recording which causes the evaluation to take more time. This is why we have only evaluated using a sample of 20 patients instead of the 500 patients we used in the 30-second overlap model. We are, therefore, less confident in the quality of our scores in the real-time model. If the findings from the previous research question results holds for low stride models, we will have a slightly better score than if we were to evaluate larger datasets.

Type	**True Positive**	**False Positive**	**False Negative**
Apnea count	6679	6282	965
% of all apneas	87.38%	82.18%	12.62%

Table 4.9: Confusion matrix from real-time detection.

In Table 4.9 we see the results from our real-time detection on 20 patients. We see that we detect 87.38 % of all apnea events that occur. This is a larger percentage of apneas in the dataset than in the 30-second evaluation, which was only 69.56%. In this table, we also see that we detect 6282 false apnea events, 397 less than the number of true positive apneas. This large percentage of false-positive apneas might be because of the dataset training with evaluations from multiple sleep technicians. As the annotations we are using for ground truth actually are human-annotated data, it might be that the apneas we are detecting as false-positive apneas might have been a true positive apnea if another technician scored the data we use for ground truth. It is, however, good that we are predicting 87.38% of all apneas and only not predicting 12.62% of them. These results are good if we want to manage the effects of apneas in real-time. Still, the severe amount of false positives might indicate that the patient has a higher apnea-hypopnea index than the patient should have for a final evaluation. One possibility in a real-world application is using this aggressive predictor for real-time data and storing all the signal data for analyzing later using 30-second strides if we want aggregate scores like AHI.

F1	**AUC**	**MCC**	**Accuracy**	**Precision**	**Recall**
0.502	0.848	0.398	0.726	0.341	0.864

Table 4.10: Scores from real-time detection.

Figure 4.3 Shows the ROC curve of the real-time detection. Table 4.10 shows that we have a higher AUC-score than the results in Table 4.7 which shows the values from the model with 30 seconds stride and a lower MCC score. The results from evaluating the model using one-second overlap seem to be less accurate predictions, but with more events correctly predicted than using a 30-second overlap.

In Figure 3.8 on page 47 we see the Yolo4Apnea real-time interface. It shows the most recent 90 seconds and the confidence the model has for an apnea event occurring at a specific time. This tool can be used by a technician to visualise the breathing leading up to each event, and the

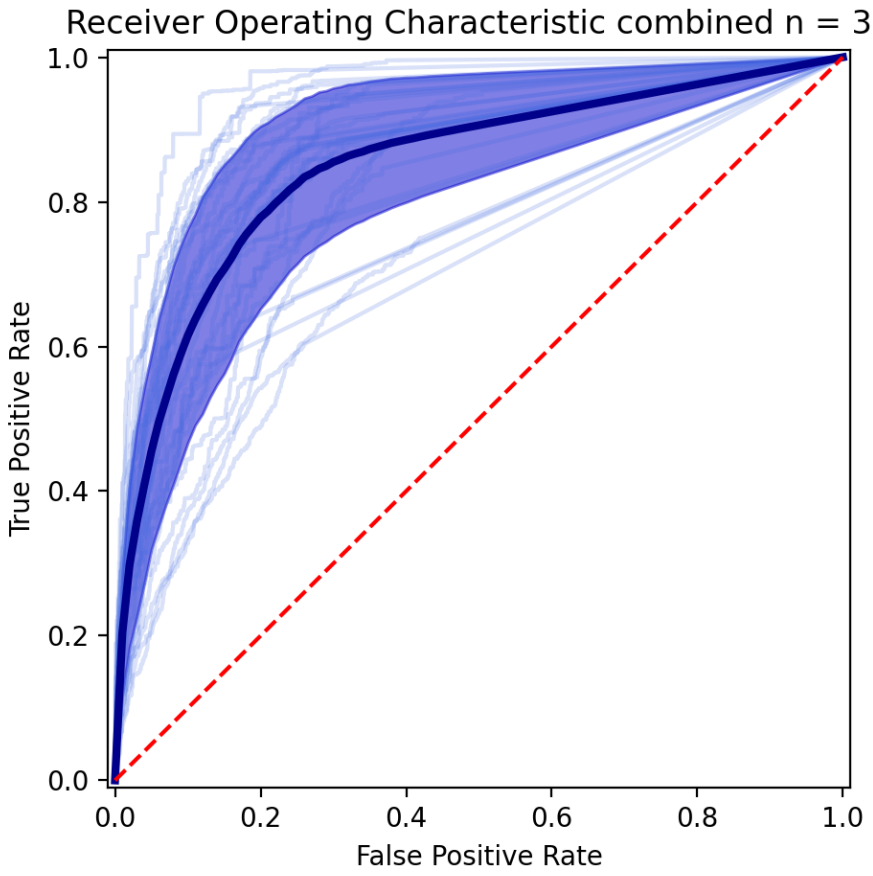


Figure 4.3: ROC curve of real-time Yolo model.

confidence the model has for this being an apnea event.

Discussion

Overlap (s)	Samples	MCC	F1	AUC	TP	FN	FP
1	20	0.398	0.502	0.848	6679	965	6282
30	500	0.432	0.548	0.784	96693	42307	71154

Table 4.11: Comparison of real-time model and model with 30 second overlap.

Table 4.11 is a table comparing the results from the previous experiments in a single table. We managed to detect parts of 87,38 of all apneas and hypopneas occurring with our model in the real-time model and 69.56% of apneas in the 30-second stride model. Looking at some other scores in Table 4.11 we see that we have a lower value in the f1-score, MCC, accuracy, precision and a higher score in AUC and recall for the real-time

model compared to the 30 second overlap model. We must have in mind that when we trained our model we search for the best hyperparameters when evaluated on a 30 second overlap model. It is probably possible to search the hyperparameter space to find parameters that give better results for real-time detection, but because of computational costs, it was not feasible to accomplish within the scope of this thesis.

RQ3: How does Yolo compare to other machine learning approaches for detecting OSA?

This is a difficult question to answer as there is a huge variance in the metrics used for evaluating a machine-learned obstructive sleep apnea detector. We have combined our results with the results from the other machine learning approaches in Table 4.12. When we explored the previous approaches, we saw that the models differed in how they reported their models performance. Some use accuracy, some specificity, some f1-score and some AUC, among other measurements. They also varied their stride when predicting, as some used a dataset that classified 30-second or 1-minute segments without overlap as either apnea or non-apnea. Our model improves this significantly as our model predicts apneas every decisecond, and each decisecond is evaluated multiple times on account of a stride value that guarantees overlap. This causes our model to grow more confident over time. Our model is also better suited for real-time detection than the other approaches because of prediction with decisecond precision.

Because of the differing ways of evaluating models, we cannot compare the scores directly, but it is interesting that our accuracy and f1-score is close to the inter-scorer accuracy and f1-score of expert annotations.

Paper	Classifier	Sen	Prec	Acc	F1	AUC
[15]	CNN SVM	74.7% 72%	74.5% 72%	74.7% 72%	74.6% 72%	
[14]	CNN LSTM CNN+LSTM					
[19]	CNN			68,75%		
[27]	LSTM(SpO2) LSTM(IHR LSTM(SpO2+IHR) LSTM(IHR)	92.9% 99.4% 84.7% 99.4%		95.5% 89% 92.%		0.98% 0.99% 0.99%
[43]	CNN(DOSED) Expert Annotations			81% 75%	57% 55%	
	Yolo 30 second overlap **Yolo 1 second overlap**		**44.5%** **34.1%**	**82.8%** **72.6%**	**54.9%** **50.2%**	**78.4%** **84.8%**

Table 4.12: Comparison of our model with the approaches from Section 2.4.

4.2 Threats to validity

We have now attempted to answer each of the research questions. We wish to mention some threats to the validity of these results, sectioned into threats to internal validity and threats to external validity

4.2.1 Threats to internal validity

The internal validity is referring to the validity of our findings regarding cause and effect.

Predicting central sleep apnea as obstructive. One threat to the internal validity is that the labels in the dataset are wrong. If many apnea events in the sleep heart health study dataset is classified wrongly then our model will predict wrong as well. Since there is no way of controlling for this we have to assume that the annotations are true.

Hyperparameter search is in a local minima. Another threat to the internal validity is that our search for parameters that results in good MCC-scores has resulted in a model that only performs well on the MCC-score. When we answered research question 2 we evaluated our model using other metrics, but we might have gotten better results if we were to have other hyperparameters.

4.2.2 Threats to external validity

The external validity is referring to validity of the results found if applied to a generalized setting.

Population selection. One of the threats to the external validity in this thesis is the population selection in the sleep heart health study dataset. The average patient was 63.1 years old and every patient was over 40 years old. No patient has history of treatment of sleep apnea, but the studies was done because patients had some problem with sleep. A threat is that our model might predict apnea events in healthy individuals as it only has trained on patients that have a high probablity of having obstructive sleep apnea. We are unsure of how well the model trained on the sleep heart health study dataset generalizes to the general populace

Variability in technician Another threat to the external validity is that the scoring done by sleep technician varies. We know sleep technician only have an average accuracy of 75 % from Thorey et al. [43]. This may cause our predictions to be incorrect.

Chapter 5

Conclusion

5.1 Summary of thesis

This thesis presents a new way of detecting obstructive sleep apnea events. We have used an object detection algorithm to detect events in a time series with good results. The use of object detection for detecting events in time series is a novel approach that has an unknown potential in further research. It seems to be performing well for detecting patterns in a time series of varying length, but the results still leave a lot to be desired.

5.2 Main results

In this thesis, we have shown that real-time detection of obstructive sleep apnea using only two sensors can be a less invasive and expensive way to test for obstructive sleep apnea. Our model is still far from being mature enough to be used in a real application but has shown promising results. Compared to other approaches, we have a more exact detector that predicts apneas with deciseconds precision and have created a tool to visualize the apneas and detections in real-time, which can be used to gain further insight into when apneas happen. Our model can detect large portions of all apneas, but also predicts many false positive apneas.

5.3 Critical assessments

This has been an extensive project for a master thesis, and the progress has been difficult to maintain when working from home without daily contact with other persons in the field. It has been a large project requiring much computational power and much storage space for training the models. I had to extract only the data I needed from the original dataset to be able to have all the data locally, which has worked well. However, it would have been an interesting experiment to train a model using only data from a single technician to see if my low scores might be affected by a high inter-scorer accuracy of the sleep technician. As I did not extract this or other

data from the large dataset, I could not do it without extracting all the data again.

I am happy to have all my experiments being fully reproducible if someone has access to the dataset. My code is available on www.github.com/shamnvik/ApneaModelPipeline [34] and can be inspected by anyone to look for errors and forked to improve it later. I have trained many models to check the results of changing hyperparameters, but I wish I could try even more to optimize the model. I believe that object detection models can be used with success on time series data with events of varying sizes, and my approach has probably only started to graze the surface of this application.

It has been difficult to compare my results against other approaches, as other approaches have used a myriad of different evaluation metrics to evaluate their models. I am confident that my choice of mainly focusing on the MCC-score is a good choice as it is a balanced measurement for highly imbalanced datasets.

The use of Yolo for detection has worked well. However, I have been hindered by not being able to use a more mature machine learning platform like TensorFlow [42] or PyTorch [29] as they have libraries that can more easily be integrated into the workflow and offers extensions that can be used for creating explainable predictions [32],[45] in a way that I do not have the capacity to do manually for Yolo. My coding skills have improved significantly during this thesis, and I have learned a lot regarding the architecture, testing and structuring of large codebases. The code has been rewritten multiple times, and the implementations have changed wildly as I ran into problems. All the experiments provided in Chapter 4 are from the final iterations of my tools.

5.4 Future work

A large part of this thesis has focused on creating a pipeline for creating training data, training, and evaluating the data, and not enough time has been spent on considering the real-world performance of the model. The thesis has focused more on applying an object detection algorithm to detect occurrences of an event in a one-dimensional array than examining the quality of such a model in a real-world application. The thesis has used data recorded from high precision hardware, but it would need to be tested on simpler low-cost hardware that the user can use in his own home if applied to real-life deployment.

Another interesting thought we have considered during this thesis is the application of having a real-time detector of sleep apnea. A CPAP is in the "on" state the whole night, but with a real-time detector, there might be a possibility to interrupt the apnea without using a CPAP which is considered quite invasive. It would have been interesting to connect the real-time detector to actuators that interrupt the apnea by shaking the bed or pillow, sound, or smell to attempt to get the patient to resume normal breathing. This, of course, needs a model that performs more accurate than

the currently developed model.

We have, throughout the project, mentioned that the search for hyperparameters is computationally expensive. We believe that a more exhaustive search could be done to find better parameters.

We also did not find time to train and evaluate models with other ways for combining the abdominal and thoracic signals. We used the sum of both values, but featurizing this by averaging the values or representing the signal values in some other way could be a vector to achieve a higher performing model. The training of our model seemed to reach a state where it was not improving with the inclusion of more batches, but it would have been interesting to train for a long time to see if the model improved since we have so much data available.

Appendix A

Results from the experiments

Table A.1 shows all relevant experiments for this thesis. It contains columns for different parameters like signal type, duration in seconds, overlap stride, apnea types, sample size, and how many batches the model was trained on. It also inclused the MCC-score, the f1-score, AUC, Recall, and the count of true positive, false negative and false positive apneas detected.

Signal	Dur	Overlap	Apnea Types	Samples	Batches	MCC	F1	AUC	Acc	Recall	TP	FN	FP
Combined	120	30	Obstructive | Hypo	20	10000	0.456	0.575	0.800	0.835	0.695	5546	2098	3697
Combined	120	30	Obstructive | Hypo	20	10000	0.456	0.575	0.800	0.835	0.695	5546	2098	3697
Combined	120	45	Obstructive | Hypo	20	10000	0.456	0.576	0.780	0.848	0.643	5183	2461	3104
Combined	120	45	Obstructive | Hypo	20	10000	0.456	0.576	0.780	0.848	0.643	5183	2461	3104
Combined	90	45	Obstructive | Hypo	20	10000	0.455	0.571	0.819	0.820	0.745	6217	1427	5129
Combined	120	60	Obstructive | Hypo	20	10000	0.455	0.577	0.770	0.853	0.616	5017	2627	2849
Combined	120	60	Obstructive | Hypo	100	10000	0.450	0.570	0.766	0.844	0.612	23696	12630	13706
Combined	120	30	Obstructive | Hypo	100	10000	0.449	0.570	0.792	0.826	0.684	26060	10266	17504
Combined	120	45	Obstructive | Hypo	100	10000	0.448	0.570	0.772	0.839	0.631	24399	11927	14744
Combined	120	15	Obstructive | Hypo	20	10000	0.445	0.560	0.819	0.810	0.754	5943	1701	4572
Combined	120	15	Obstructive | Hypo	100	10000	0.436	0.557	0.809	0.802	0.738	27825	8501	21390
Combined	120	30	Obstructive | Hypo	500	10000	0.432	0.548	0.784	0.828	0.666	96693	42307	71154
Combined	120	45	Obstructive | Hypo	500	10000	0.432	0.548	0.765	0.841	0.614	90149	48851	59770
Abdominal	90	45	Obstructive | Hypo	20	10000	0.428	0.568	0.777	0.829	0.660	5854	1790	4546
Combined	150	45	Obstructive | Hypo	20	10000	0.423	0.528	0.695	0.871	0.430	3295	4349	1030
Combined	60	45	Obstructive | Hypo	20	10000	0.421	0.541	0.815	0.794	0.757	6721	923	7612
Combined	120	60	Central | Hypo	100	10000	0.401	0.499	0.740	0.838	0.577	19035	12846	15412
Combined	120	1	Obstructive | Hypo	20	10000	0.398	0.502	0.848	0.726	0.864	6679	965	6282
Thoracic	90	45	Obstructive | Hypo	20	10000	0.387	0.520	0.752	0.820	0.609	5099	2545	3942
Combined	180	45	Obstructive | Hypo	20	10000	0.370	0.457	0.650	0.866	0.330	2410	5234	586
Combined	30	45	Obstructive | Hypo	20	10000	0.274	0.413	0.662	0.803	0.428	5280	2364	6374
Combined	30	45	Obstructive | Hypo	20	10000	0.274	0.413	0.662	0.803	0.428	5280	2364	6374
Combined	120	60	Obstructive | Hypo	100	10000	0.269	0.367	0.602	0.847	0.236	9113	27213	2974
Combined	30	45	Obstructive | Hypo	500	10000	0.249	0.386	0.644	0.800	0.392	90460	48538	120452

Table A.1: Comparison of all experiments

List of Figures

List of Tables

Bibliography

[1] *3.3. Metrics and scoring: quantifying the quality of predictions*. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (visited on 13/05/2021).

[2] Alexey. *AlexeyAB/darknet*. original-date: 2016-12-02T11:14:00Z. 11th May 2021. URL: https://github.com/AlexeyAB/darknet (visited on 12/05/2021).

[3] Kathleen Askland et al. 'Educational, supportive and behavioural interventions to improve usage of continuous positive airway pressure machines in adults with obstructive sleep apnoea'. In: *Cochrane Database of Systematic Reviews* 4 (2020). ISSN: 1465-1858. DOI: 10.1002/14651858.CD007736.pub3. URL: https://www-cochranelibrary-com.ezproxy.uio.no/cdsr/doi/10.1002/14651858.CD007736.pub3/full (visited on 09/05/2021).

[4] Adam V Benjafield et al. 'Estimation of the global prevalence and burden of obstructive sleep apnoea: a literature-based analysis'. In: *The Lancet Respiratory Medicine* 7.8 (1st Aug. 2019), pp. 687–698. ISSN: 2213-2600. DOI: 10.1016/S2213-2600(19)30198-5. URL: http://www.sciencedirect.com/science/article/pii/S2213260019301985 (visited on 15/04/2020).

[5] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao. 'YOLOv4: Optimal Speed and Accuracy of Object Detection'. In: *arXiv:2004.10934 [cs, eess]* (22nd Apr. 2020). arXiv: 2004.10934. URL: http://arxiv.org/abs/2004.10934 (visited on 13/04/2021).

[6] Sabri Boughorbel, Fethi Jarray and Mohammed El-Anbari. 'Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric'. In: *PLoS ONE* 12.6 (2nd June 2017). ISSN: 1932-6203. DOI: 10.1371/journal.pone.0177678. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5456046/ (visited on 15/04/2021).

[7] Christen Brownlee. 'A slumber not so sweet: Loss of brain cells that regulate breathing may cause death during sleep'. In: *Science News* 168.7 (2005), pp. 102–102. ISSN: 1943-0930. DOI: https://doi.org/10.2307/4016651. URL: https://onlinelibrary.wiley.com/doi/abs/10.2307/4016651 (visited on 11/04/2021).

[8] Ramiro Casal, Leandro Di Persia and Gastón Schlotthauer. 'Sleep-wake stages classification using heart rate signals from pulse oximetry'. In: *Heliyon* 5 (1st Oct. 2019), e02529. DOI: 10.1016/j.heliyon.2019.e02529.

[9] James Chen. *Neural Network Definition*. Investopedia. URL: https://www.investopedia.com/terms/n/neuralnetwork.asp (visited on 12/04/2021).

[10] Marc Claesen and Bart De Moor. 'Hyperparameter Search in Machine Learning'. In: *arXiv:1502.02127 [cs, stat]* (6th Apr. 2015). arXiv: 1502.02127. URL: http://arxiv.org/abs/1502.02127 (visited on 27/04/2021).

[11] Facebook and Twitter. *Overnight Sleep Study Testing: What to Expect*. Verywell Health. URL: https://www.verywellhealth.com/what-to-expect-in-a-sleep-study-3015121 (visited on 07/05/2021).

[12] Kunihiko Fukushima. 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'. In: *Biological Cybernetics* 36.4 (1st Apr. 1980), pp. 193–202. ISSN: 1432-0770. DOI: 10.1007/BF00344251. URL: https://doi.org/10.1007/BF00344251 (visited on 13/04/2021).

[13] Glos Martin et al. 'Characterization of Respiratory Events in Obstructive Sleep Apnea Using Suprasternal Pressure Monitoring'. In: *Journal of Clinical Sleep Medicine* 14.3 (), pp. 359–369. DOI: 10.5664/jcsm.6978. URL: https://jcsm.aasm.org/doi/10.5664/jcsm.6978 (visited on 08/05/2021).

[14] Maziar Hafezi et al. 'Sleep Apnea Severity Estimation from Respiratory Related Movements Using Deep Learning'. In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). ISSN: 1558-4615. July 2019, pp. 1601–1604. DOI: 10.1109/EMBC.2019.8857524.

[15] Rim Haidar, Irena Koprinska and Bryn Jeffries. 'Sleep Apnea Event Detection from Nasal Airflow Using Convolutional Neural Networks'. In: *Neural Information Processing*. Ed. by Derong Liu et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 819–827. ISBN: 9783319701394. DOI: 10.1007/978-3-319-70139-4_83.

[16] Sondre Hamnvik, Pierre Bernabé and Sagar Sen. 'Yolo4Apnea: Real-time Detection of Obstructive Sleep Apnea'. In: Twenty-Ninth International Joint Conference on Artificial Intelligence. Vol. 5. 9th July 2020, pp. 5234–5236. DOI: 10.24963/ijcai.2020/754. URL: https://www.ijcai.org/proceedings/2020/754 (visited on 08/05/2021).

[17] *How Long Does a Sleep Study Last?* Advanced Sleep Medicine Services, Inc. 25th Oct. 2016. URL: https://www.sleepdr.com/the-sleep-blog/how-long-does-a-sleep-study-last/ (visited on 07/05/2021).

[18] Vanessa Ibáñez, Josep Silva and Omar Cauli. 'A survey on sleep assessment methods'. In: *PeerJ* 6 (25th May 2018). ISSN: 2167-8359. DOI: 10.7717/peerj.4849. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5971842/ (visited on 11/04/2021).

[19] Syed M.S. Islam et al. 'Deep Learning of Facial Depth Maps for Obstructive Sleep Apnea Prediction'. In: *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*. 2018 International Conference on Machine Learning and Data Engineering (iCMLDE). Dec. 2018, pp. 154–157. DOI: 10.1109/iCMLDE.2018.00036.

[20] Tsung-Yi Lin et al. 'Microsoft COCO: Common Objects in Context'. In: *arXiv:1405.0312 [cs]* (20th Feb. 2015). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312 (visited on 22/02/2021).

[21] Stephen Marsland. *Machine Learning: An Algorithmic Perspective, Second Edition*. Routledge & CRC Press. URL: https://www.routledge.com/Machine-Learning-An-Algorithmic-Perspective-Second-Edition/Marsland/p/book/9781466583283 (visited on 12/04/2021).

[22] *Matplotlib: Python plotting — Matplotlib 3.4.1 documentation*. URL: https://matplotlib.org/ (visited on 08/05/2021).

[23] *numpy.array — NumPy v1.20 Manual*. URL: https://numpy.org/doc/stable/reference/generated/numpy.array.html (visited on 07/05/2021).

[24] *Obstructive Sleep Apnea: Gadgets and Devices Guide*. WebMD. URL: https://www.webmd.com/sleep-disorders/sleep-apnea/sleep-apnea-gadgets (visited on 18/05/2021).

[25] *OpenCV: YOLO DNNs*. URL: https://docs.opencv.org/master/da/d9d/tutorial_dnn_yolo.html (visited on 08/04/2021).

[26] S. R. Pandi-Perumal, D. Warren Spence and Ahmed S. BaHammam. 'Polysomnography: An Overview'. In: *Primary Care Sleep Medicine: A Practical Guide*. Ed. by James F. Pagel and S. R. Pandi-Perumal. New York, NY: Springer, 2014, pp. 29–42. ISBN: 9781493911851. DOI: 10.1007/978-1-4939-1185-1_4. URL: https://doi.org/10.1007/978-1-4939-1185-1_4 (visited on 08/05/2021).

[27] Rahul Krishnan Pathinarupothi et al. 'Single Sensor Techniques for Sleep Apnea Diagnosis Using Deep Learning'. In: *2017 IEEE International Conference on Healthcare Informatics (ICHI)*. 2017 IEEE International Conference on Healthcare Informatics (ICHI). Aug. 2017, pp. 524–529. DOI: 10.1109/ICHI.2017.37.

[28] Jean Louis Pépin et al. 'Side Effects of Nasal Continuous Positive Airway Pressure in Sleep Apnea Syndrome: Study of 193 Patients in Two French Sleep Centers'. In: *Chest* 107.2 (1st Feb. 1995), pp. 375–381. ISSN: 0012-3692. DOI: 10.1378/chest.107.2.375. URL: https://www.sciencedirect.com/science/article/pii/S0012369216349650 (visited on 09/05/2021).

[29] *PyTorch*. URL: https://www.pytorch.org (visited on 07/05/2021).

[30] Joseph Redmon and Ali Farhadi. 'YOLO9000: Better, Faster, Stronger'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). ISSN: 1063-6919. July 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[31] Joseph Redmon et al. 'You Only Look Once: Unified, Real-Time Object Detection'. In: *arXiv:1506.02640 [cs]* (9th May 2016). version: 4. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640 (visited on 13/04/2021).

[32] *Responsible AI Toolkit*. TensorFlow. URL: https://www.tensorflow.org/responsible_ai?hl=nb (visited on 08/05/2021).

[33] Michael J. Sateia. 'International Classification of Sleep Disorders-Third Edition'. In: *CHEST* 146.5 (1st Nov. 2014), pp. 1387–1394. ISSN: 0012-3692. DOI: 10.1378/chest.14-0970. URL: https://journal.chestnet.org/article/S0012-3692(15)52407-0/abstract (visited on 06/04/2020).

[34] *shamnvik/ApneaModelPipeline*. GitHub. URL: https://github.com/shamnvik/ApneaModelPipeline (visited on 14/05/2021).

[35] Leming Shi et al. 'The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models'. In: *Nature Biotechnology* 28.8 (Aug. 2010), pp. 827–838. ISSN: 1546-1696. DOI: 10.1038/nbt.1665. URL: https://www.nature.com/articles/nbt.1665 (visited on 15/04/2021).

[36] Shreyak. *Building a Convolutional Neural Network (CNN) Model for Image classification.* Medium. 8th June 2020. URL: https://becominghuman.ai/building-a-convolutional-neural-network-cnn-model-for-image-classification-116f77a7a236 (visited on 13/04/2021).

[37] *Sleep Heart Health Study - 04 Dataset Introduction - Sleep Data - National Sleep Research Resource - NSRR*. URL: https://sleepdata.org/datasets/shhs/pages/04-dataset-introduction.md (visited on 22/02/2021).

[38] *Sleep Heart Health Study - 05 Polysomnography Introduction - Sleep Data - National Sleep Research Resource - NSRR*. URL: https://sleepdata.org/datasets/shhs/pages/05-polysomnography-introduction.md (visited on 22/02/2021).

[39] *Sleep Heart Health Study - 08 Equipment Shhs1 - Sleep Data - National Sleep Research Resource - NSRR*. URL: https://sleepdata.org/datasets/shhs/pages/08-equipment-shhs1.md (visited on 22/02/2021).

[40] *Sleep Heart Health Study - Readme - Sleep Data - National Sleep Research Resource - NSRR*. URL: https://sleepdata.org/datasets/shhs/pages/README.md (visited on 22/02/2021).

[41] American Academy of Sleep Medicine. *The International Classification of Sleep Disorders, Revised*. 26th July 2011. URL: https://web.archive.org/web/20110726034931/http://www.esst.org/adds/ICSD.pdf (visited on 09/04/2021).

[42] *TensorFlow*. TensorFlow. URL: https://www.tensorflow.org/?hl=nb (visited on 07/05/2021).

[43] Valentin Thorey et al. 'AI vs Humans for the diagnosis of sleep apnea'. In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). ISSN: 1558-4615. July 2019, pp. 1596–1600. DOI: 10.1109/EMBC.2019.8856877.

[44] David Troy. *Sleep Study*. Sleep Education. URL: https://sleepeducation.org/patients/sleep-study/ (visited on 07/05/2021).

[45] *Using TensorFlow with Explainable AI | AI Platform (Unified)*. Google Cloud. URL: https://cloud.google.com/ai-platform-unified/docs/explainable-ai/tensorflow?hl=nb (visited on 08/05/2021).

[46] *What is Supervised Learning?* 2nd Apr. 2021. URL: https://www.ibm.com/cloud/learn/supervised-learning (visited on 12/04/2021).