Czech University of Life Sciences Prague

Faculty of Economics and Management

Informatics

Master's Thesis

Methods of AI in Classification of Antepartum Heart Rate DeepNN CAD system

Mohamed Darwish

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Bc. Mohamed Darwish

Informatics

Thesis title

Methods of AI in Classification of Antepartum Heart Rate

Objectives of thesis

The main objective is to construct a classifier of antepartum heart rate using methods of artificial intelligence.

To achieve this goal, following milestones have been set:
• write an overview of classification algorithms, where they can be applied, and how to implement them using Python programming language,
• demonstrate the use of selected algorithm on a case study focusing on Antepartum Hearth Rate classification,
• describe the services offered by cloud vendors that can help in provisioning such a model, such as cloud computing and data storage, and propose a suitable architecture for a classifier application.

Methodology

The methodology of the thesis is based on analysis and study of the relevant technical and scientific sources focusing on fundamental AI models that can be used for data classification, with specific attention paid to models used for heart rate classification. Based on synthesis of gained knowledge a classification case study focusing on antepartum heart rate will be implemented using standard methods of software engineering and artificial intelligence. The paper will also focus on the infrastructure needed to implement the model, and how to leverage the cloud services in order to have an end to end classification service that could be potentially used by practitioners and doctors.

The proposed extent of the thesis

60-80 pages

Keywords

Machine learning, artificial intelligence, data classification, heart rate

Recommended information sources

ALARSAN, F.I., YOUNES, M. Analysis and classification of heart diseases using heartbeat features and machine learning algorithms. J Big Data 6, 81 (2019). https://doi.org/10.1186/s40537-019-0244-x

AZIZ, S., AHMED, S. & ALOUINI, MS. ECG-based machine-learning algorithms for heartbeat classification. Sci Rep 11, 18738 (2021). https://doi.org/10.1038/s41598-021-97118-5

RUSSELL, Stuart J., Peter NORVIG a Ernest DAVIS. Artificial intelligence: a modern approach. 4th ed. Upper Saddle River: Prentice Hall, 2020. ISBN 1292401133

TETTAMANZI, A. – TOMASSINI, M. *Soft computing : integrating evolutionary, neural, and fuzzy systems.* Berlin: SPRINGER, 2001. ISBN 3-540-42204-8.

Expected date of thesis defence

2022/23 SS – FEM

The Diploma Thesis Supervisor

Ing. Petr Hanzlík, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 13. 3. 2023

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 13. 3. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 30. 03. 2023

Declaration

I declare that I have worked on my master's thesis titled "Methods of AI in Classification of Antepartum Heart Rate" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague on date of submission _____

Acknowledgement

I would like to express my gratitude for whomever supported my journey through my masters degree, including my supervisor, my professors, my classmates and my friends, also special thanks to my family for their continuous support and guidance through my entire learning journey.

Methods of AI in Classification of Antepartum Heart Rate

Abstracts

Continuous monitoring of the fetal heart rate (FHR) signal has been widely used to allow obstetricians to obtain detailed physiological information about newborns. The objective of this thesis is to examine a novel approach of analyzing FHR signals with the help of AI and propose a framework that can help obstetricians make better decisions. This work is composed of mainly two parts in which the first part showcases the use of a CNN model to help classify the FHR signals into normal and pathological categories while the second part discusses the potential of integrating the CNN model with the already existing medical infrastructure. We proposed transforming the signals into 2D images using wavelet transformation and feeding it into a deep CNN model. The model was able to achieve a specificity of 84% and sensitivity of 30%. We then discussed the potential of utilizing the model in the cloud to turn it into a reliable CAD system.

Keywords: Machine learning, artificial intelligence, data classification, fetal heart rate, hypoxia, signal processing, wavelet transformation, wavelet scalogram, convolution neural network, cloud technologies.

Metody AI v klasifikaci předporodní srdeční frekvence

Abstrakt

Kontinuální monitorování signálu srdeční frekvence plodu (FHR) bylo široce používáno, aby porodníci mohli získat podrobné fyziologické informace o novorozencích. Cílem této práce je prozkoumat nový přístup k analýze signálů FHR pomocí AI a navrhnout rámec, který může porodníkům pomoci k lepšímu rozhodování. Tato práce se skládá převážně ze dvou částí, v nichž první část představuje použití modelu CNN pro pomoc při klasifikaci signálů FHR do normálních a patologických kategorií, zatímco druhá část pojednává o potenciálu integrace modelu CNN s již existující lékařskou infrastrukturou. Navrhli jsme transformaci signálů do 2D obrazů pomocí vlnkové transformace a jejich zavedení do hlubokého modelu CNN. Model byl schopen dosáhnout specificity 84 % a senzitivity 30 %. Poté jsme diskutovali o potenciálu využití modelu v cloudu, abychom z něj udělali spolehlivý CAD systém.

Klíčová slova: Strojové učení, umělá inteligence, klasifikace dat, srdeční frekvence plodu, hypoxie, zpracování signálu, vlnková transformace, vlnkový skalogram, konvoluční neuronová síť, cloudové technologie.

I. Introduction

Fetal heart activity and fetal heart rate monitoring are crucial in ensuring the health and wellbeing of a fetus during pregnancy. Cardiotocography (CTG) is a common method used to monitor fetal heart activity and rate by measuring the electrical signals from the fetal heart and the uterine contractions of the mother. CTG can provide important information about the fetus's condition, including signs of fetal distress or acidosis.

Monitoring fetal heart rate (FHR) and uterine contractions (UC) allows obstetricians to identify potential cases of fetal hypoxia, which can occur even in an otherwise uncomplicated pregnancy. Although a fetus has its own inherent ability to manage insufficient oxygen levels during delivery, prompt intervention may be necessary in certain situations to prevent negative outcomes, in which hypoxia, with prevalence lying in the region of 0.6% to 3.5%, is one of them and considered to be the third most common cause of newborn death [1].

CTG

Cardiotocography, commonly abbreviated as CTG, is a non-invasive diagnostic tool used in obstetrics to monitor fetal wellbeing during pregnancy and labor. The procedure involves simultaneously recording the fetal heart rate (FHR) and uterine contractions using an electronic fetal monitor. CTG monitoring provides valuable information about fetal oxygenation and fetal heart activity, helping obstetricians to identify potential signs of fetal distress or hypoxia. The data obtained from CTG monitoring is used to inform decisions regarding fetal management, such as the need for timely intervention or delivery.

CTG is typically performed by placing two sensors on the mother's abdomen: an ultrasound sensor to measure fetal heart rate and a tocodynamometer to measure uterine contractions. However, in some cases, a scalp electrode may be used to more accurately measure the fetal heart rate. This involves attaching a small electrode to the fetal scalp, which transmits the fetal heart rate to the CTG machine. The scalp electrode is inserted into the vagina during labor and delivery, and the electrode is attached to the fetal scalp via a small puncture.

CTG is widely used by obstetricians especially during the last phases of labor and right before baby delivery, however, according to a study held by *Lavender et al. (2011)* [2] in which included over 8,000 women with low-risk pregnancies who were randomized to receive either CTG monitoring or intermittent auscultation - *a method widely used before CTG* - during labor found no significant differences in the incidence of adverse neonatal outcomes between the two groups, suggesting that intermittent auscultation may be as effective as CTG monitoring in low-risk pregnancies suggesting that there may not be significant improvements in delivery outcomes with CTG monitoring compared to intermittent auscultation in low-risk pregnancies. Other publications also like the one published by *Alfirevic and Devane (2017)* which is a systematic review that evaluates the effectiveness of CTG monitoring for fetal assessment in pregnancy and labor. The authors reviewed 133 randomized controlled trials that compared CTG monitoring to intermittent auscultation or other forms of fetal monitoring. The review found that CTG monitoring did not reduce the risk of perinatal death or cerebral palsy compared to intermittent auscultation [3].Moreover, CTG is considered the main suspect of the increased rate of cesarean sections for objective reasons [1], even though *Lavender et al.*[2] suggested that this may be due to the increased detection of fetal heart rate abnormalities with CTG monitoring, leading to more interventions, however, *Alfirevic and Devane*[3] found that the use of CTG monitoring was associated with an increased risk of instrumental deliveries, cesarean deliveries, and neonatal admissions to intensive care due to the detection of fetal heart rate abnormalities that do not necessarily indicate fetal distress.

FIGO

	Normal ✓	Suspicious ?	Pathological !
Baseline	110-160 bpm	Lacking at least one characteristic of normality, but with no pathological features	<100 bpm >10 mins
Variability	5-25 bpm		• Reduced variability < 5bpm for >50 minutes. • Increased variability >25bpm for >50 minutes. • Sinusoidal pattern for >30minutes.
Decelerations	No repetitive* decelerations		Repetitive* late or prolonged (>3minutes) decelerations for >30minutes (or >20minutes if reduced variability). Isolated deceleration >5minutes
Interpretation	No hypoxia/acidosis	Low probability of hypoxia/acidosis	High probability of hypoxia/acidosis
Clinical Management	No intervention necessary to improve fetal oxygenation state	Action to correct reversible causes if identified, close monitoring or adjunctive methods	Immediate action to correct reversible causes, adjunctive methods, or if this is not possible expedite delivery. In acute situations immediate delivery should be accomplished.

Table 1: Adapted from FIGO, 2015: CTG classification criteria and recommended management

In 1986, FIGO developed guidelines (seen in Table 1) to improve the accuracy of cardiotocography by examining the macroscopic morphological features of fetal heart rate (FHR) and their relationship with topographic measurements. Despite the availability of these guidelines for more than two decades, interpreting CTG results remains a persistent challenge [1]. Studies conducted by *Blix et al. (2003)* [4] and *Ayres-de-Campos et al. (1997)* [5] have both explored the variability in CTG interpretation among different and even the same observers. *Blix et al. (2003)* [4] recruited 58 obstetricians and midwives to assess 30 CTGs and found significant variations in interpreting FHR features. Meanwhile, *Ayres-de-Campos et al. (1997)* [5] observed moderate to substantial differences in the interpretation of 20 CTGs by 24 experienced obstetricians over two weeks. The level of agreement between the same observer was only moderate to substantial, and there was significant variability in the interpretation of features such as baseline FHR, variability, and decelerations among different

observers. These findings underline the need for more objective and standardized CTG interpretation methods, including exploring the feasibility of automated FHR evaluation. This paper aims to contribute to this discourse.

Challenges

The driving force behind this paper stemmed from an enduring sense of detachment between scholarly works and real-world healthcare settings.*Topol et al. (2020)* [6] identified concerns among clinicians and patients regarding the reliability and safety of AI systems used for FHR analysis. One of these concerns is the lack of transparency and explainability of AI algorithms, which can make it challenging for stakeholders to trust and rely on these systems. In addition, the accuracy and reliability of AI algorithms can be affected by the quality and representativeness of the data used to train them. Clinicians and patients may therefore be skeptical of AI systems that have not been validated on diverse populations. Finally, stakeholders may be concerned about the role of AI in clinical decision-making, particularly if they believe it could replace the judgment and expertise of human clinicians or be used to make decisions without adequate input or explanation from the clinician.

In their article, *Sikdar and Mukhopadhyay (2021)*[7] explore the challenges associated with incorporating AI into healthcare systems, specifically in the area of obstetrics and gynecology. They emphasize that one major obstacle is the incompatibility of legacy systems with AI applications, which can impede the integration of these technologies into clinical practice. The authors stress the importance of overcoming these technical barriers in order to fully realize the potential benefits of AI in this field. In addition to the technical challenges, [7] also discusses other obstacles to the adoption of AI in clinical practice. One such obstacle is resistance to change, particularly among healthcare providers who may be hesitant to adopt new technologies that disrupt established workflows or pose unfamiliar risks. This reluctance can slow the adoption of AI in the field of obstetrics and gynecology. The authors also highlight the critical role of data quality and standardization. For AI algorithms to function effectively, they require high-quality and standardized data. However, healthcare data is often

incomplete, inconsistent, and difficult to access, making it challenging to develop reliable and accurate AI algorithms.

II. Objectives and Methodology

Methodology

As we mentioned earlier, implementing an AI solution to any area in the medical field is challenging and filled with a lot of obstacles. Rather than focusing on a single AI model or a certain algorithm we instead propose an entire framework or architecture that could be easily implemented and integrated with the current medical infrastructure to help assist doctors and obstetrician make a better decisions regarding data collected during first and second stage of labor which makes them interpret FHR signals in a more robust and reliable way.

Vision

Our proposal aims to demonstrate the potential benefits of using a computer-aided decision system to assist obstetricians in analyzing fetal heart rate (FHR) signals for early detection of hypoxia or acidosis. The timely detection of these conditions is essential for preventing adverse outcomes in newborns. The detection of hypoxia and fetal acidosis can vary from case to case, with some instances being detected several hours or days before delivery, while in others, it may be identified shortly before delivery. Therefore, it is imperative to have a reliable system that analyzes the cardiotocography (CTG) signals obtained a few hours before delivery, enabling prompt medical interventions when required.This paper will focus on the different elements of creating such a CAD system rather than just exploring ML/deep learning models.

Objectives

This paper aim to contribute toward different objectives;

Data Collection and Review

One aspect we will delve into is the data collection process and how we can bring together data from various sources of CTG devices such as scalp electrodes and ultrasound sensors. We

will also explore the CTU-HUB database used to analyze the FHR signals. We also propose a framework that aims to have a single, authoritative source of information that can serve as a foundation for any future efforts to build a more dependable system.

Data Preprocessing

Creating a CAD system involves various complexities, particularly in the selection of appropriate AI models and their corresponding preprocessing techniques. Each AI model has unique requirements and preprocessing steps, and thus it is important to explore and discuss various approaches to determine the advantages and disadvantages of each.

Data Modeling

Several attempts have been made to model FHR signals, which can be categorized into two groups: researches that utilize deep learning and other researches that use traditional machine learning methods after extracting multiple features from the signals. Our primary goal is to develop a deep learning CNN model that is comparable to the clinical benchmark and can be considered reliable. The model will be fed with 2D images produced from the continuous wavelet transform of the 1D FHR signals.

Infrastructure

Our plan is to present a practical implementation of the proposed CAD system and outline the necessary steps to integrate it into the existing infrastructure of any Hospital. This will involve proposing a roadmap for the successful deployment of the system. We will also discuss the potential benefits that such a system can bring to the hospital and its patients.

III. Literature Review

CTU-HUB

In this paper, we utilized the CTUHUB database which contains 552 intrapartum fetal heart rate (FHR) recordings. These recordings were acquired over a period of two years from April 2010 to August 2012 at the obstetrics ward of the University Hospital located in Brno, Czech Republic. The OB TraceVue® system was used to store all recordings in electronic form. Further information regarding the database used can be found in [8]. To summarize, the

authors selected the recordings from a pool of 14,492 deliveries that were monitored using fetal monitors such as STAN S21 and S31 from Neoventa Medical and Avalon FM40 and FM50 from Philips Healthcare. All CTG signals were stored electronically in a proprietary format in the OB TraceVue® system by Philips and were later converted into a text format using proprietary software also provided by Philips [8].

Out of those deliveries, authors of [8] slimmed down the dataset to fit certain criterias:

- 871 non-singleton pregnancies were excluded
- 7637 excluded for insufficient information (missing CTG records or pH values
- 4866 excluded for clinical reasons (ie: premature, maternal age < 18, etc.)
- 552 were chosen out of the remaining 1118
-

The decision to use the CTU_HUB database was based on its suitability for various automated approaches. The authors further filtered the recordings to only include those that had umbilical artery pH data available. This specific selection was made as it served as the target variable for the convolutional neural network (CNN) model training process. Each recording contains the fetal heart rate (FHR) signals with a sampling rate of 4 Hz. The length of each signal may vary for each case, but all signals cover a minimum period of 30 minutes before delivery.

Umbilical Artery pH

Multiple publications have shown that umbilical artery pH is a reliable measure of neonatal respiratory hypoxia. For example, *Ramanah et al. (2017) [9]* discussed the use of umbilical artery pH in fetal heart rate monitoring to assess fetal well-being during labor. The authors highlighted that umbilical artery pH can be used to identify neonates at risk for hypoxic-ischemic encephalopathy, a condition caused by inadequate oxygen supply to the brain during labor and delivery.Another study by *Kashanian et al. (2012) [10]* investigated the association between umbilical artery pH and neonatal morbidity and mortality in preterm deliveries. The authors found that low umbilical artery pH values were significantly associated with an increased risk of adverse neonatal outcomes, such as respiratory distress syndrome,

intraventricular hemorrhage, and neonatal death. Hence the decision was made that records with pH < 7.15 are pathological [11].

Signal Processing

Signal processing is an essential part of preparing data for use in deep learning models. The raw signals that are acquired often contain unwanted noise, artifacts, or missing values, which can significantly impact the performance of a model. Various signal processing techniques such as filtering, normalization, and interpolation are used to remove noise and artifacts, fill in the missing values, and standardize the signal characteristics. By applying these preprocessing steps, we can not only improve the quality of the data but also extract meaningful features from the signal that are relevant to the task at hand. Consequently, processing the data before using it in a deep learning model is critical to ensuring that the model is trained on high-quality data and can learn and generalize well to new data.

Noise in a signal (seen in Figure 1) refers to the presence of unwanted or unpredictable variations that are not related to the primary signal being measured. These fluctuations can originate from various sources, including external interference, technical constraints of the measuring devices, or problems with signal transmission.Normalization and interpolation techniques are used to denoise the signal.

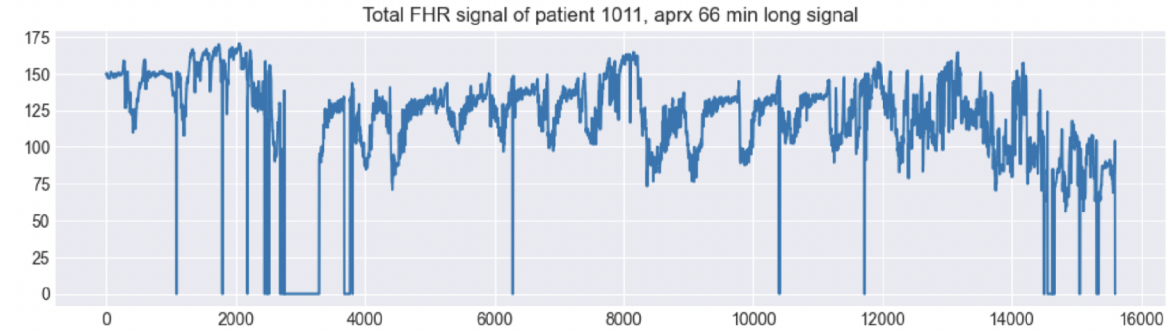


Figure 1: Noise appears as spontaneous spikes quite visible especially at the end of the signal (few minutes before delivery) such noise is caused by the movement of the fetus and also caused by the medical team to deliver the baby.

In contrast, artifacts are distortions or anomalies in the signal introduced by the measurement process itself, including factors such as electrode placement or motion artifacts.We defined an artifact as a signal loss for over than 15 seconds as seen in Figure 2

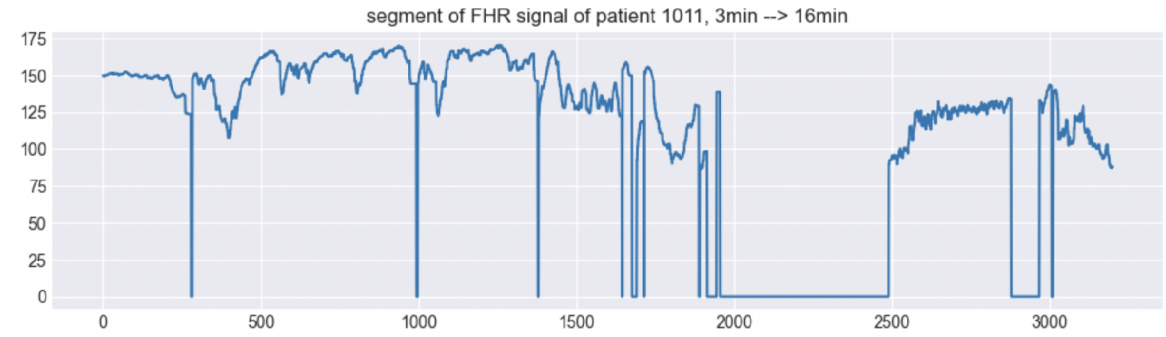


Figure 2: A Segment taken from the total signal to emphasize the presence of artifacts seen at samples from 2000 until 2500.

Effective signal processing can mitigate the negative effects of both noise and artifacts, and there are various techniques available to remove these unwanted components from the signal. The choice of signal processing techniques depends on the specific characteristics of the signal and the source of the unwanted components, as well as the desired characteristics of the final processed signal. *Zhao, Zhidong, et al [12]* presented a novel approach for classifying FHR signals that involved the extraction of features from both the time and frequency domains. To extract features from the frequency domain, the authors utilized the Discrete Wavelet Transform (DWT), while statistical measures like mean, variance, skewness, and kurtosis were used to extract features from the time domain. This approach allowed for a comprehensive and multifaceted analysis of the FHR signals, enabling the accurate classification of the signals for improved diagnosis and patient management.

*Zhao et al. (2019)*in *[13] - which this paper is heavily inspired by* - converted FHR signals into images using Continuous Wavelet Transform (CWT). The use of CWT allows for the extraction of time-frequency information from the signal, which can be represented as images. The resulting images were then processed using various image processing techniques to improve the contrast and remove noise. The preprocessed images were then used to train a convolutional neural network (CNN) to detect fetal hypoxia. By using CWT and image

processing techniques, this approach allowed for the extraction of more complex features from the FHR signals, which improved the accuracy of the classification model. The authors suggest that this approach has the potential for developing non-invasive methods for detecting fetal hypoxia. In our paper, it is decided that the denoised signals are transformed into 2D images obtained from the CWT of the original signal.

Continuous Wavelet Transform

FHR signals are inherently noisy and irregular which at the same time have a certain structure. One way to characterize this phenomena mathematically and quantify the structure found in the FHR signal is by transforming the signal from the time domain to the time-frequency domain using CWT.

Continuous Wavelet Transform (CWT) is a powerful mathematical tool that enables the analysis of signals with time-varying frequency components. It has found applications in various fields such as engineering, physics, and biology. The CWT uses a wavelet function that is scaled and shifted over the entire signal to obtain a time-frequency representation of the signal. This transformation allows the identification of local features in the signal, including the identification of transient events and the detection of changes in frequency content.

The CWT is a time-frequency analysis technique that is different from the traditional Fourier transform, which provides only frequency domain information of a signal. The CWT can provide more detailed information about the time-frequency content of a signal, making it a valuable tool in the analysis of non-stationary signals, where the frequency content of the signal changes over time. Moreover, the CWT can be used to extract features from signals that are relevant to specific applications, such as the detection of anomalies, the identification of patterns, and the classification of signals. Despite its usefulness, the CWT has some limitations, such as its computational complexity and the selection of an appropriate wavelet function for a given signal. Nevertheless, the CWT has become a standard tool for signal analysis and has been applied in various fields such as speech recognition, image processing, and biomedical signal processing. In this paper, CWT is used to represent the FHR signal from the time domain to the time-frequency domain to extract frequencies contributing to the signal.

Wavelets

Representing a signal in the frequency domain through Fourier transform can present certain challenges. While it allows access to information about the frequencies present in the signal, it comes at the cost of losing temporal information. Fourier transform compresses the signal in time to extract frequency components, as the Fourier function only considers frequency and not the timing of frequency occurrence. This trade-off between time and frequency information is highlighted by the Heisenberg uncertainty principle.

To address this limitation, wavelets offer an alternative approach that builds upon Fourier transform while making necessary modifications. Unlike Fourier transform, which breaks the signal down into a sum of sine and cosine functions, wavelet transformation involves analyzing functions called wavelets that are temporally restrained. Wavelets are characterized by their short-lived oscillations that are confined to a specific time period. Figure 3 depicts the set of functions that meet two primary constraints.

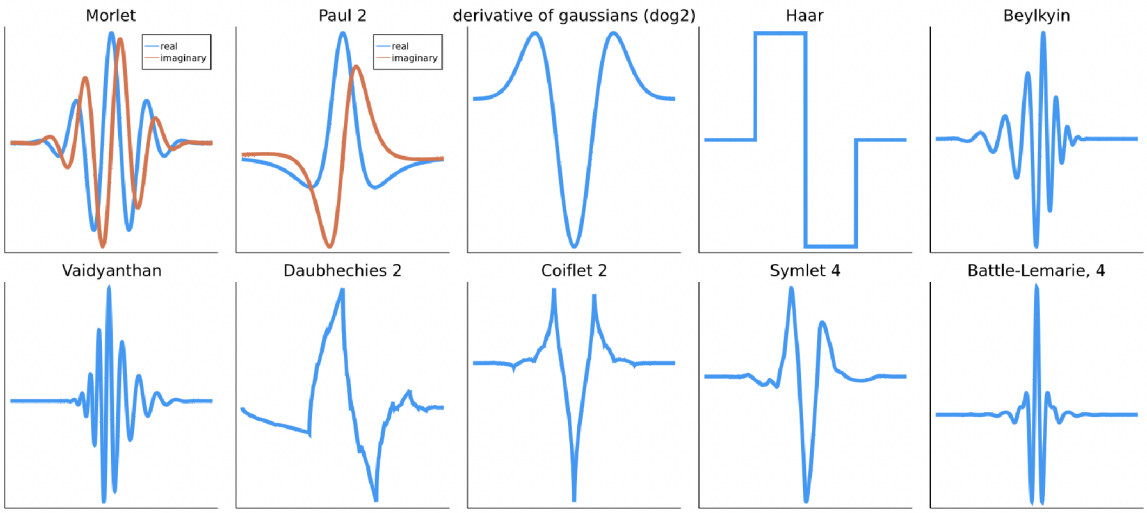


Figure 3: Different wavelets families and their representation

Admissibility: The wavelet must exhibit zero-mean behavior, meaning that it oscillates between positive and negative values over time while maintaining a finite energy. This property ensures that the wavelet can be reliably employed in signal analysis tasks. In other

words, the integral or area under the curve of the wavelet function should add up to zero, as shown in equation (1). This condition can also be expressed as the wavelet function having no zero-frequency component.

$$\int_{-\infty}^{+\infty} \Psi(t)\, dt \;=\; 0 \quad (1)$$

Localization: Another key requirement for a function to be considered a wavelet is that it must demonstrate localization properties in both time and frequency domains. Specifically, the wavelet should be well-concentrated in time, implying that it has a short support or duration in the time domain. Additionally, the wavelet should exhibit a band-limited frequency representation, meaning that most of its energy is concentrated within a specific frequency range. By satisfying these constraints, the wavelet can effectively capture local characteristics of the signal in both the time and frequency domains. This localization property is particularly useful in signal analysis, as it allows the wavelet to extract precise details from a signal at different time scales.To put it simply the wavelet needs to have finite energy, where the energy of the wavelet is defined as:

$$\int_{-\infty}^{+\infty} |\Psi(t)|^2\, dt \;<\; \infty \quad (2)$$

Morlet Wavelet

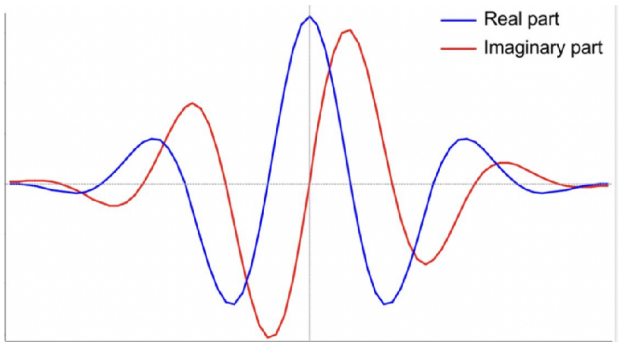


Figure 4: The Morlet wavelet representation in both the real and imaginary part of the function Notice that the Morlet function satisfies both admissibility and localization constraints.

The Morlet function is a complex-valued wavelet that is well-suited for analyzing oscillatory signals, such as those found in FHR signals. The Morlet wavelet has a Gaussian shape in the time domain and is modulated by a complex sinusoid in the frequency domain. This allows it to provide a high degree of localization in both domains, making it a valuable tool in analyzing local features of FHR signals. The Morlet wavelet has been widely used in various studies on FHR signal analysis, such as in detecting and classifying fetal distress and predicting fetal acidemia. For example, in a study by *Casas et al. (2018)*, the Morlet wavelet was utilized in analyzing FHR signal features to identify patterns indicative of fetal acidemia [14]. Similarly, in a study by *Zhu and Liu (2021)*, the Morlet wavelet was employed in FHR signal analysis to detect and classify fetal distress[15]. These studies demonstrate the utility of the Morlet wavelet in FHR signal analysis. The decision on choosing the Morlet wavelet for our analysis was based on the fact that the Morlet can produce a complex wavelet transform, this complex wavelet transform can capture the amplitude and phase information of the signal, which is particularly useful for analyzing signals that exhibit non-linear and non-stationary properties.The real component of the Morlet function is defined as:

$$\Psi(t) = k_0 \cdot cos(wt) \cdot e^{\frac{-t^2}{2}} \quad (3)$$

In which the basis function is a cosine wave with a certain frequency that corresponds to our FHR signal frequency normalized with constant k_0 and dampening it with the exponent factor. Typically during Fourier transform, the time representation of the signal $y(t)$ which is one dimensional is transformed into its respective frequency domain representation $\hat{y}(f)$ which is also one dimensional, the key difference that separates wavelets from fourier transform is that the same one dimensional signal is now represented as two dimensional surface where one axis representing frequency and the other axis representing time; where $y(t) \rightarrow T(t, f)$. Hence the value of $T(t, f)$ represents the contribution of the frequency f at time t. The construction of such function T is done by constructing daughter wavelets, seen in equation (4) by scaling and translating the Morlet wavelet where a and b represent the scaling and translating factors of the mother wavelet. It is also worth noting the existence of a relation between the scaling

factor and the frequency of the mother wavelet function, and since such relation exists, the function T could be represented as a function of a and b.

$$\Psi_{a,b} = \Psi\left(\frac{t-b}{a}\right) \quad (4)$$

That indicates that value of $T(a, b)$ is equal to the contribution of $\psi_{a,b}$ to the original signal.

$$T(a, b) = \int_{-\infty}^{+\infty} y(t) \cdot \psi_{a,b}(t)\, dt \quad (5)$$

It also reflects the similarity between the original signal and the newly constructed daughter wavelet scaled and translated. The procedure of varying the values of the scaling factor and calculating the integral is called convolution where the variation of the scaling factor will pull out other frequency components that contributed at a specific time to the original signal and to see which frequencies are more prominent at that time point. The value of the dot product in the above formula indicates the similarity of the daughter wavelet $\psi_{a,b}$ with the original FHR signal $y(t)$, this value will be close to zero when the original FHR signal frequency is significantly different than the daughter wavelet which is expected, however the value could also be close to zero even if $y(t)$ and $\psi_{a,b}$ have the same frequency but phase shifted due to the translation factor, and since it is the objective of the wavelet transformation is to measure the frequency as a function of time it is important to introduce the imaginary part of the Morlet wavelet which make us redefine the Morlet wavelet as:

$$\Psi(t) = k \cdot e^{iw_0 t} \cdot e^{\frac{-t^2}{2}} \quad (6)$$

The key concept is to calculate the convolution of the signal with both real and imaginary parts, then our convolution function for a fixed wavelet scale will map the translation parameter b to the point in the complex plane where the real component is the value of the convolution at that time point with the real part of the wavelet and the imaginary component is the value of the convolution with the imaginary part of the wavelet. The power of the frequency or the intensity of its contribution at each point in time is given by the distance from the resulting point to the origin which is also known as the absolute value of the complex

number.As before varying the scaling factor allows us to analyze the FHR signal at different frequencies.

Wavelet Scalogram

The resulting complex function $T(a, b)$ can be represented as a two dimensional image known as the wavelet scalogram, seen in Figure 9. The image represents the values obtained from the wavelet transformation of a signal.The time axis represents the location of the signal in time, while the frequency axis represents the frequency content of the signal. The intensity of the color at each point in the two dimensional plane corresponds to the power of the wavelet frequency at that point.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of artificial neural network that is widely used in image recognition and classification tasks. They work by processing input data through a series of convolutional and pooling layers to extract features, followed by fully connected layers for classification.In the context of analyzing fetal heart rate (FHR) signals, we can take the images produced from the scalogram of the wavelet transformation of the signals and use them as inputs for a CNN. The CNN can then be trained to recognize patterns in the images and make predictions about the underlying FHR signals. The activation function used in most CNNs is the Rectified Linear Unit (ReLU).

ReLU is a simple non-linear function that sets all negative values to zero and leaves all positive values unchanged. It is computationally efficient and has been shown to improve the performance of CNNs in many image recognition tasks. CNNs consist of multiple layers, each with its own set of trainable parameters. The layers are typically connected in a feedforward manner, meaning that the output of one layer is used as the input to the next layer.

The first layer in a CNN is a convolutional layer, which applies a set of filters to the input image to extract features such as edges, corners, and textures. The filters are learned during the

training process, and their values are optimized to maximize the network's ability to recognize patterns in the input data.

The next layer in a CNN is typically a max pooling layer, which reduces the spatial dimensions of the feature maps produced by the convolutional layer. This reduces the computational cost of the network and helps to prevent overfitting. The max pooling layer works by dividing the feature map into non-overlapping regions and taking the maximum value in each region.

Another important layer in a CNN is the dropout layer, which helps to prevent overfitting by randomly dropping out a fraction of the neurons in the network during training. This forces the network to learn more robust representations of the input data by preventing it from relying too heavily on any one set of features.We propose a simple architecture consisting of seven distinct layers:

Input Layer: In this paper, the continuous wavelet transform is used to convert the original 1D time series into a 2D image as the input layer of the CNN, the image is cropped and all three channels are used as an input.

Convolution Layer: A Convolutional Neural Network (CNN) is a type of Deep Neural Network (DNN) that uses a specialized convolutional structure to minimize memory consumption and the number of parameters in the network. In the convolution layer, the hidden layers are interconnected via a feature map, which extracts pixel-level abstracted features through convolution operations using one or more convolution kernels (also known as filters). Each convolution kernel scans the entire feature map using a sliding window mechanism, gathering and fusing information from each small area to create a partial representation of the input image's features.One of the key benefits of using a CNN is the parameter sharing mechanism, which ensures the filter parameters used in each convolutional layer remain consistent. This approach ensures that image content remains unaffected by location, which can significantly reduce the number of optimization parameters. This

parameter sharing mechanism is a crucial and attractive aspect of the CNN algorithm, enabling it to extract complex image features with minimal computational resources.

Pooling Layer: To enhance the effectiveness of the Convolutional Neural Network (CNN) model, a pooling layer, also referred to as a sub-sampling layer, is usually incorporated periodically between consecutive convolution layers.This layer exploits the fact that certain image features that are valuable in one area may also be relevant in other areas. By grouping semantically similar features, the pooling operation compresses the convolution output's eigenvectors and the number of parameters, resulting in a less complex model with faster computation times that is less prone to overfitting.The pooling operation is performed by mapping features in each sub-region of the input feature map using a stride-based approach, similar to the convolution layer. The most commonly used pooling methods are max pooling, average pooling, and randomized pooling. In the current CNN model, max pooling is utilized to calculate the maximum value of the image area as the pooled result. This approach results in the selection of the most dominant features, leading to more effective feature extraction and classification.

Fully-connected Layer: This layer produces high-level features of the input image that are statistically analyzed using a classifier to calculate the probability of the corresponding class label. Following several rounds of convolution and pooling, the input image information is abstracted into more information-dense features. The convolution and pooling layers are necessary for automatic image feature extraction, while the fully-connected layer is responsible for the final classification task once the feature transformation is complete.By utilizing the fully-connected layer, the model is able to leverage the extracted features to accurately classify input images. As the final stage of the network, this layer takes advantage of the higher-level features that have been extracted through the previous layers to perform the classification task. Through this process, the CNN model is capable of effectively handling complex image classification tasks.

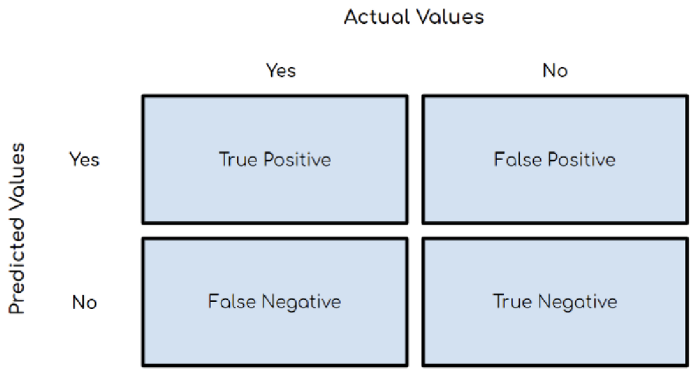
Drop-out Layer: When it comes to classification tasks, preventing overfitting is of utmost importance. Overfitting occurs when a model achieves high accuracy on the training data, but

fails to generalize well to new, unseen data. This phenomenon arises when the model memorizes the noise in the training data instead of learning the underlying patterns. To tackle this problem, various techniques have been proposed in the literature, and [16, 17] suggest multiple solutions. The dropout layer is usually placed after the fully-connected layer. During the training process, several neural units are randomly dropped from the network with a certain probability. This technique enables the model to become more robust and less dependent on specific neurons, leading to better generalization performance [13].

Classification Layer: Finally softmax max is used to separate the output classes.

Evaluation Metrics

The Confusion matrix is an evaluation matrix that is more suitable in classification problems, as it provides a more insightful picture which is not only the performance of a predictive model but also which classes are being predicted correctly.



Where:

- **True Positive (TP)**: The model correctly predicts the positive class.
- **False Positive (FP)**: The model incorrectly predicts the positive class.
- **True Negative (TN)**: The model correctly predicts the negative class.

- **False Negative (FN)**: The model incorrectly predicts the negative class.

We can extract insights from the confusion matrix like the sensitivity (also called the true positive rate) which measures the proportion of positive cases that are correctly identified, as well as the specificity (also called true negative rate) which measures the proportion of negative cases that are correctly identified. It is vital to measure both sensitivity and specificity of the model along with the accuracy to have a better picture regarding the predictive performance and what to expect.

Infrastructure and service architecture

Cloud Services and Cloud Computing

As we mentioned in our objectives and methodology, our purpose in this paper is not only providing a model to classify the FHR signals but also to propose a framework that utilizes the model and turn it into an end to end service that can be integrated with the current medical systems, and such we will be leveraging the cloud technologies to achieve that.

The cloud, in its simplest form, refers to a collection of remote servers that are accessed over the internet. These servers can be used for a variety of purposes, including storage, computing, and data management. Cloud computing is the process of utilizing the cloud servers to deliver on-demand computing resources, while cloud storage is the ability to store data on the cloud servers rather than on local devices. Cloud computing and storage have become increasingly popular due to their scalability, accessibility, and cost-effectiveness. The cloud can provide a powerful infrastructure to support computationally intensive tasks such as machine learning algorithms, including CNNs. One of the most significant advantages of the cloud is its scalability. Cloud resources can be easily scaled up or down to meet changing demands, making it an ideal platform for handling large amounts of data or computing-intensive tasks.
This scalability is particularly beneficial for CNNs, as these models require large amounts of computational power and data storage. Additionally, the cloud offers high availability and reliability, which ensures that data and applications are accessible 24/7, from any location.The

cloud has numerous uses in the medical field, including but not limited to, patient data management, telemedicine, and medical imaging.

The cloud can also be used to store and manage vast amounts of patient data securely, which can be accessed by authorized medical personnel from any location.Medical imaging is another area where the cloud can be utilized to store and manage large image data sets, which can then be accessed and analyzed by healthcare professionals. Cloud-based medical imaging solutions can help to improve patient outcomes by enabling more efficient diagnosis and treatment.

Cloud Computing

When it comes to training a CNN to classify FHR signals, there are a number of challenges that must be addressed. One of the main challenges is the large amount of data that must be processed and analyzed. This can be a time-consuming process and requires a lot of computational power. Another challenge is the need for specialized hardware and software to train the CNN effectively. These challenges can be particularly daunting for smaller organizations or research teams that may not have access to the necessary resources. However, cloud computing can help overcome many of these challenges. By utilizing cloud-based services for computing and storage, organizations can access virtually unlimited computational resources and storage capacity. This can greatly reduce the time and cost associated with training a CNN, as well as eliminate the need for specialized hardware and software. Additionally, cloud computing can provide greater flexibility and scalability, allowing organizations to easily increase or decrease resources as needed to meet their specific needs. For example, the computing resources could be scaled both up and out during the training process of the CNN model and then scaled back down when the model is trained, saving money and ensuring availability of resources when needed.

Cloud Storage

When it comes to creating and saving scalogram images from FHR signals, there are a number of challenges that must be addressed. One of the main challenges is the large amount of data that must be processed and stored. FHR signals can be quite long and can produce a large

number of scalogram images, which can quickly consume significant amounts of storage space. Additionally, these images must be stored in a way that allows for easy retrieval and analysis, which can be a complex task. Again by utilizing cloud-based storage services, organizations can store large amounts of data without having to invest in any hardware or software.

Additionally, cloud storage services typically offer features such as redundancy, backup, and recovery, which can help ensure that data is safe and accessible at all times. Furthermore, cloud storage can provide greater flexibility and accessibility, allowing organizations to easily access and share data from anywhere in the world. This can be particularly useful for research teams or organizations with multiple locations or remote staff. Additionally, cloud storage solutions often offer integration with other cloud-based services, such as computing and analytics tools, which can further enhance the efficiency and effectiveness of data analysis.

Cloud Data Streaming

Data streaming is a technology that allows data to be transferred in real-time from one system to another. With the advent of cloud computing, data streaming has become an increasingly popular method for processing and analyzing large amounts of data. Cloud-based data streaming systems, such as Amazon Kinesis, Google Cloud Dataflow, and Microsoft Azure Stream Analytics, provide a scalable and cost-effective way to handle the massive volumes of data generated by modern applications. These systems allow data to be ingested from a variety of sources, including IoT devices, social media platforms, and enterprise systems, and processed in real-time to generate insights and support decision-making.

The benefits of data streaming in the cloud are numerous, including real-time processing of data, scalability, fault-tolerance, and cost-effectiveness. Real-time processing allows organizations to gain insights and make decisions based on the most up-to-date data, while scalability and fault-tolerance ensure that the system can handle fluctuations in data volume and maintain high levels of availability. Additionally, cloud-based data streaming systems often offer pay-as-you-go pricing models, which can significantly reduce the cost of data processing compared to on-premise solutions. Which is extremely useful for our problem here

as monitoring FHR signals is a time sensitive activity and the room for making a decision regarding those signals is usually small.

ETL

ETL stands for extract, transform, and load, and it refers to the process of moving data from one or more sources, transforming it into a format that can be easily analyzed and loaded into a target system. The main purpose of ETL is to ensure that the data is accurate, consistent, and reliable.

The extract phase involves gathering data from different sources such as databases, flat files, and APIs. The data can be structured or unstructured, and it can come from various types of data sources. The data is then transformed into a format that is suitable for analysis. This process involves data cleaning, data validation, data enrichment, and data aggregation. The load phase involves loading the transformed data into the target system, such as a data warehouse, a database, or a cloud storage service. The target system is optimized for data analysis and can store large amounts of data for a long period of time.

ETL has become increasingly important in recent years as organizations have started to accumulate vast amounts of data from multiple sources. ETL is crucial in enabling organizations to harness the full potential of their data by ensuring that the data is accurate and consistent. Amazon Lambda for instance is a powerful tool that can be used to help with the ETL process. serverless compute service provided by cloud providers allows developers to run code without provisioning or managing servers. When it comes to ETL, such solutions can be used to transform, clean, and process data as it is being ingested into a database or data warehouse. AWS Lambda functions for example can also be triggered by various events such as data being uploaded to Amazon storage or data being ingested into Amazon data streaming service. Once triggered, Lambda can perform ETL transformations on the data in real-time, and store the processed data into a database such as Amazon Aurora.

OB TraceVue®

Before introducing the proposed architecture, it is important to provide an overview of the architecture utilized to collect the data used in this study. As outlined in section III, the FHR signals were obtained using the OB TraceVue® system, which was responsible for capturing and storing the signals in its database. By understanding how the fetal heart rate monitoring system operates and using the OB TraceVue® as a point of reference, we aim to propose cloud-based solutions that can optimize and enhance the reliability of the system.

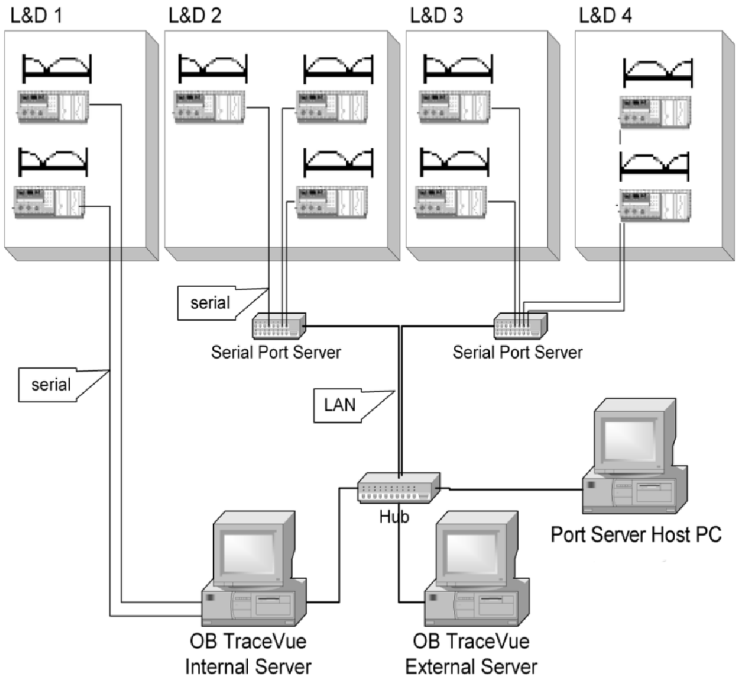


Figure 5: Example of the OB TraceVue® system installation, in which several monitors are connected to multiple servers needed for the system to operate.

The system above in Figure 5 contains multiple components necessary for any other FHR systems. From a hardware perspective, the OB TraceVue system consists of bedside fetal monitors, a central monitoring station, and three servers. The fetal monitors are connected to the mother's abdomen using transducers that measure fetal heart rate (FHR), uterine activity, and maternal vital signs such as blood pressure and pulse. The monitors transmit the signals

wirelessly to the central monitoring station, where the data is displayed on a screen for review by clinicians. The central monitoring station also includes a keyboard and mouse for user input, as well as a printer for generating reports. The server component of the OB TraceVue system stores all patient data and provides advanced features such as archiving, data retrieval, and remote access. The server is typically installed in a dedicated server room or data center and is connected to the central monitoring station and other workstations using a local area network (LAN). The system also supports remote access, allowing authorized users to view patient data from outside the hospital network. Based on the documentation provided publicly by the OB TraceVue, the system includes three servers:

- OB TraceVue internal server - This server is responsible for storing patient data and providing advanced features such as archiving, data retrieval, and remote access. The internal server is connected to the central monitoring station and other workstations using a local area network (LAN).

- OB TraceVue external server - This server is used to provide remote access to the OB TraceVue system. Authorized users can connect to the external server from outside the hospital network to view patient data and perform various tasks such as patient admission, transfer, and discharge.

- Port server host PC - This server is used to manage communication between the fetal monitors and the internal server. The host PC acts as a bridge between the fetal monitors and the internal server, allowing data to be transmitted wirelessly from the monitors to the server.

In the upcoming sections, we will provide a detailed overview of the proposed CAD system. This includes an explanation of how data will be streamed from the OB TraceVue database, which is hosted on the internal server, as well as a description of how the CNN model will be trained and utilized.

IV. Practical Part

Signal denoising

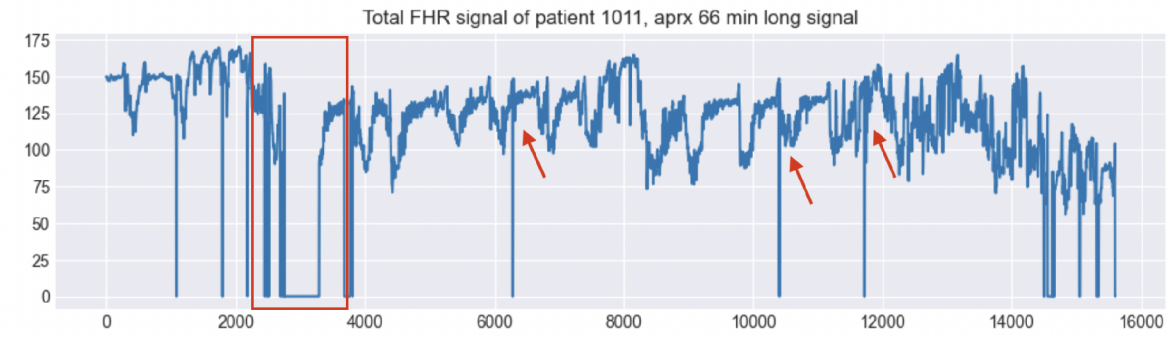


Figure 6: Noisy Signal, artifact is present and highlighted by the red box along with some visible noise also highlighted by the red arrow.

To ensure high-quality data for our CNN model, we applied a series of denoising methods to the original signals obtained from the database. As suggested in [13], we utilized these techniques to remove unwanted noise and artifacts that could negatively impact the performance of our model. A gap(G) in the signal is defined as the period where the FHR reading is equal zero, long gaps where G>15 seen in Figure 2 were completely removed from the signal as where gaps are < 15 were filled by using rolling mean and lastly cubic spline interpolation is applied to replace the extreme (not physiological) values (< 50 bpm and > 200 bpm)[13]. The rolling mean and spline interpolation were done using the SciPy library in Python 3.10

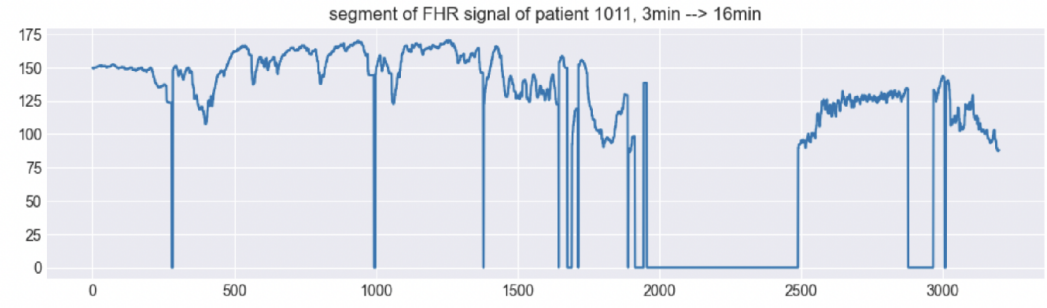


Figure 7: Top to bottom; shows the preprocessing steps proposed in [13] starting by removing the artifacts, signal denoising by interpolation, the final denoised signal is presented in the last image.

NOTE:Those steps were implemented on the entire signal.The figures are just for illustrative purposes to showcase the outcome of each preprocessing step individually.

Python was used to clean the data as described above along with its libraries like numpy.

```python
def remove_long_gaps(signal, threshold, freq=FREQ):
    """
    Removing gaps if they last for more than the threshold
    A gap is defined as a period where the signal is zero for a period of time

    Args:
        signal (numpy.ndarray): 1D NumPy array containing the signal data.
        threshold (int): Time in seconds; a gap longer than the threshold will be removed

    Returns:
        numpy.ndarray: 1D NumPy array of the signals with the gaps removed
    """
    threshold = threshold * freq
    signal_cleaned = np.array(list(chain(*(l for k,g in groupby(signal)if len(l:=list(g))<threshold or k))))
    return signal_cleaned
```

```python
def fill_zeros_with_rolling_mean(signal, window_size):
    """
    Fill zero values in a signal with the rolling mean of the previous
    non-zero values using a window of specified size.

    Args:
        signal (numpy.ndarray): 1D NumPy array containing the signal data.
        window_size (int): Size of the rolling window to use.

    Returns:
        numpy.ndarray: 1D NumPy array with zero values replaced by the rolling mean.
    """
    rolling_window = window_size
    corrected_fhr = signal.copy() # Make a copy to avoid modifying input signal
    zero_idxs = np.where(signal == 0)[0]
    for i in zero_idxs:
        prev_vals = corrected_fhr[max(0, i-rolling_window):i]
        next_vals = corrected_fhr[i+1:min(i+rolling_window+1, len(signal))]
        avg = np.mean(np.concatenate((prev_vals, next_vals)))
        corrected_fhr[i] = avg
    return corrected_fhr
```

```python
def remove_artifacts(signal, window_size):
    """
    Replace data values less than 50 and greater than 200 in a signal with the rolling mean of the previous
    non-zero values that are between 50 and 200 using a window of specified size.

    Args:
        signal (numpy.ndarray): 1D NumPy array containing the signal data.
        window_size (int): Size of the rolling window to use in seconds.

    Returns:
        numpy.ndarray: 1D NumPy array with out-of-range values replaced by the rolling mean.
    """
    rolling_window = window_size * FREQ
    corrected_signal = signal.copy() # Make a copy to avoid modifying input signal
    for i, val in enumerate(corrected_signal):
        if val < 50 or val > 200:
            prev_vals = [x for x in corrected_signal[max(0, i-rolling_window):i] if 50 <= x <= 200]
            next_vals = [x for x in corrected_signal[i+1:min(i+rolling_window+1, len(signal))] if 50 <= x <= 200]
            if prev_vals or next_vals:
                avg = np.mean(prev_vals + next_vals)
                corrected_signal[i] = int(avg)
    return corrected_signal
```

Snippet 1: From top to bottom are three snippets representing the preprocessing steps needed to clean the data

The code here is responsible for cleaning the entire FHR signal and producing a clean signal ready for scalogram as seen in the last image in Figure 7.

Continuous Wavelet Transform

As we mentioned before the one dimensional signal is then represented in the time-frequency domain and transformed into a two dimensional image using the scalogram, as seen in Figure 9. After denoising and cleaning the signal, we utilize the PyWavelets library in python to perform the continuous wavelet transformation.Pywavelets library is an open source wavelet transform software for Python. It combines a simple high level interface with low level C and Cython performance.It allowed us to choose the mother wavelet we need, in our case it is the Morlet wavelet, also it allowed us to define the scales of the mother wavelet.

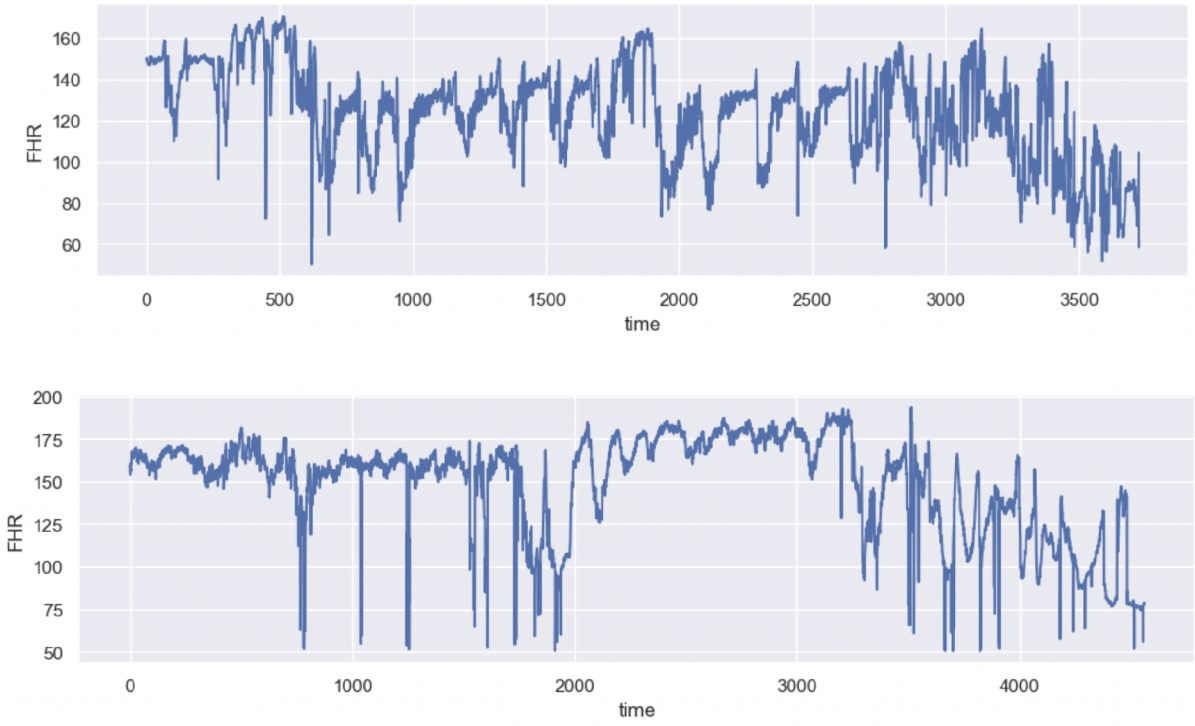


Figure 8: Patient id 1011 (Normal) FHR signal (TOP), patient id 1017 FHR signal (BOTTOM)

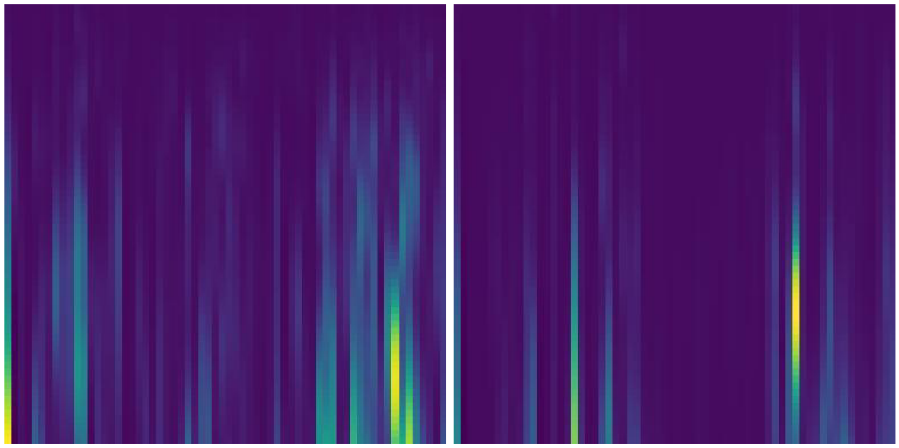


Figure 9: The scalogram image of the CWT, the scales are ranging from 10 to 100 and the power of the frequency and its contribution to the original signal is highlighted by the contrast in the image.

Patient id 1011 (Normal) scalogram image (Left), patient id 1017 scalogram image (Right)

When analyzing FHR signals with CNNs using wavelet transforms, the selection of appropriate wavelet scales depends on a variety of factors. These include the frequency content of the signal, the desired balance between accuracy and computational complexity, and empirical observations. It is important to carefully consider these factors when choosing the range of scales to use in order to achieve optimal performance in a given application.In [5] the authors used scales ranging from 2 to 32. They chose this range of scales because it allowed them to capture both low and high frequency components of the FHR signal, and because it provided a good trade-off between accuracy and computational complexity. However authors of [19] used scales ranging from 4 to 128. They chose this range because it allowed them to capture a wide range of frequency components, and because it provided better performance compared to using a smaller or larger range of scales. We are suggesting using scales ranging from 10 to 100, this will allow us to capture a wide range and at the same time limit the computational complexity.

```python
def _cwt_image(signal, wavelet='morl', scales=np.arange(10, 100)):
    coef, freqs = pywt.cwt(signal, scales, wavelet, sampling_period=1/4) #4Hz
    power = (np.abs(coef)) ** 2
    return power[:, :len(signal)], freqs
```

Snippet 2: A python 3.10 function which uses the PyWavelets library to calculate the CWT of the FHR signal

As described in section two, the Morlet wavelet contains both real and imaginary parts, thus the coefficients of the wavelet transform are complex number where the absolute value of the coefficient is called the magnitude and represents the strength of the contribution of the corresponding wavelet at a given scale and location. Squaring the magnitude gives the power of the wavelet contribution.Therefore, in Snippet 2, the line

*power = (np.abs(coef)) ** 2*

calculates the power of the wavelet coefficients obtained from the CWT of the input signal. This power is a measure of the energy distribution of the signal in the time-frequency plane, and it is commonly used to create scalogram images that visualize the signal's time-frequency characteristics.Taking the absolute value of the coefficients before squaring ensures that the power values are always positive. Additionally, the squaring operation amplifies the magnitudes, which can help in highlighting the important features in the scalogram image.

```python
def create_image(signal, path):
    signal_cf, _ = _cwt_image(signal)
    img = Image.fromarray(signal_cf)
    img_resized = img.resize((64, 64)).convert('RGB')
    plt.imshow(img_resized)
    img_resized.format = 'JPEG'
    img_resized.save(path, 'JPEG')

def plot_cwt_img(signal, path):
    signal_cf, freqs = _cwt_image(signal)

    # Resize the image to 64x64 pixels using PIL
    img = Image.fromarray(signal_cf)
    img_resized = img.resize((64, 64))

    # Convert the PIL image back to a numpy array
    signal_resized = np.asarray(img_resized)

    # Plot the resized image with the original colormap
    fig, ax = plt.subplots(figsize=(5, 5))
    ax.imshow(signal_resized, cmap='viridis', aspect='auto', interpolation='none')
    ax.set_xlabel('Time (s)')
    ax.set_ylabel('Frequency (Hz)')

    plt.axis('off')
    plt.savefig(path, bbox_inches='tight', pad_inches = 0)
#     plt.show()
    plt.close(fig)
```

Snippet 3: A python function used to create the scalogram image out of the coefficients obtained from the CWT

The following step will be representing the power of the signal into scalogram images and since the CNN model expects fixed shape of input, all images are resized into 64x64 and saved locally to be used as the training and testing data for our proposed CNN model. It is generally a good idea to resize the images produced from the scalogram, but it is important to carefully consider the trade-offs between computational cost and model performance. Resizing images can help reduce the computational cost and memory requirements of the model, as well as provide a consistent input size to the model. In addition, resizing can help reduce the effects of image distortions caused by differences in the input signal, such as variations in signal amplitude or duration. On the other hand, resizing the images can also result in the loss of important information and details in the image. This can lead to a decrease in the model's ability to distinguish between normal and pathological signals, ultimately reducing sensitivity. Additionally, resizing the images can also introduce unwanted artifacts or distortions that can negatively impact the model's performance. However due to computation limitations resizing the images was a must in our case.

```
count = 0
for file in os.listdir(DATA_PATH):
    # Looping through all signals
    if file.endswith(".hea"):
        # Preprocessing of removing long gaps, rolling mean for short gaps, removing noise
        signal_path               = Path(file).stem
        signal, fields            = wfdb.rdsamp(f'ctu-hub/{signal_path}',sampfrom=0, channels=[0])
        signal_gap_removed        = remove_long_gaps(signal, 15)
        signal_rm                 = fill_zeros_with_rolling_mean(signal_gap_removed, 5)
        signal_artifactes_removed = remove_artifacts(signal_rm, 10)
        signal                    = signal_artifactes_removed.flatten()
        # Creating images and saving them into training and testing directories
        if df_patients.loc[int(signal_path)]['target']:
            if int(signal_path) in training_pos_ids:
                plot_cwt_img(signal, PATH_output_training+f"/{signal_path}_{1}.jpg")
            else:
                plot_cwt_img(signal, PATH_output_testing+f"/{signal_path}_{1}.jpg")
        else:
            if int(signal_path) in training_neg_ids:
                plot_cwt_img(signal, PATH_output_training+f"/{signal_path}_{0}.jpg")
            else:
                plot_cwt_img(signal, PATH_output_testing+f"/{signal_path}_{0}.jpg")
        count += 1
        print(f'file {signal_path}: Transformation completed, length = {len(signal)}')
        print(f'{count} files completed')
```

Snippet 4: A for loop iterating over all signals, cleaning and preprocessing it and then converting it into a 64x64 image.

Modeling

The images are split into training and testing sets, where the testing set consists of 20% of the entire dataset. The entire dataset is split in a way so that both training and testing sets contain the same percentage of positive to negative classes. Shuffling the training data before feeding it into a Convolutional Neural Network (CNN) model is a crucial step in preventing the model from relying on any patterns or sequences that may exist in the data due to its order. Failure to shuffle the data could result in the model learning to recognize specific patterns or sequences in the data instead of learning the essential underlying patterns relevant to the task at hand. This overfitting can lead to poor performance on new, unseen data. Shuffling the data removes any potential biases resulting from the order of the examples. It randomly mixes up the data, so the model sees a diverse range of examples during training. Consequently, the model can learn the underlying patterns relevant to the task, without being biased by the order of the data. Overall, shuffling the training data helps the CNN model to better generalize its learned patterns to new and unseen data, thus improving its generalization performance. This is why

shuffling training data before passing it to a CNN model is an essential and common practice in machine learning.

```
# Shuffling the data
idx = np.random.permutation(len(X_train))
X_train = X_train[idx]
y_train = y_train[idx]
```

```
print(f'X_train shape: {X_train.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'X_test shape : {X_test.shape}')
print(f'y_test shape : {y_test.shape}')
```

```
X_train shape: (373, 32, 32, 3)
y_train shape: (373, 2)
X_test shape : (179, 32, 32, 3)
y_test shape : (179, 2)
```

Snippet 5: Shuffling the training data, while also printing out the shape of the labels and images set.

As we discussed in section III the CNN model consists of multiple layers and those layers are defined using python libraries Keras. Keras is a powerful library that is capable of doing complex computations running on top of Tensorflow, which is another deep learning library.Both libraries are used to create the model seen in Snippet 6.

```
# Define model
model = models.Sequential()
model.add(layers.BatchNormalization(momentum=0.75, epsilon=1e-3))


# Add convolution layer
model.add(layers.Conv2D(15, (5, 5), strides=(1,1), padding='valid'))
model.add(layers.ReLU())

# Add normalization layer
model.add(layers.BatchNormalization(momentum=0.75, epsilon=1e-3))

# Add convolution layer
model.add(layers.Conv2D(8, (3, 3), strides=(1,1), padding='valid'))
model.add(layers.ReLU())

# Add normalization layer
model.add(layers.BatchNormalization(momentum=0.75, epsilon=1e-3))

# Add pooling layer
model.add(layers.MaxPooling2D((2, 2), strides=(2,2), padding='valid'))

# Add fully-connected layer
model.add(layers.Flatten())
# Add dropout layer
model.add(layers.Dropout(0.5))
# Add classification layer
model.add(layers.Dense(2, activation="softmax"))
```

Snippet 6: The CNN is defined using Keras, the model consists of multiple layers and the output layer uses softmax activation to decide on the output class.

The code defines a convolutional neural network (CNN) model that consists of several layers. The first layer is a BatchNormalization layer, which normalizes the input data, helping to improve the training and generalization performance of the model.Next, the model has two consecutive Conv2D layers with ReLU activation functions.The ReLU activation function is added after each Conv2D layer, which helps to introduce non-linearity into the model. This nonlinearity is essential because FHR data is often non-linear in nature, and non-linear activation functions allow the model to better capture these nonlinear relationships.

The first Conv2D layer has 15 filters of size (5, 5), while the second Conv2D layer has 8 filters of size (3, 3). These layers learn and extract features from the input data using convolutional operations. The use of multiple Conv2D layers allows the model to learn more complex and abstract representations of the input data, this is because each layer can learn to recognize different features at different levels of abstraction, building on the previous layer's

learned representations.After the second Conv2D layer, another BatchNormalization layer is added to normalize the output of the previous layer.

Normalizing the activations of the model's layers helps to prevent vanishing or exploding gradients, which can slow down or prevent the model from learning.Next, a MaxPooling2D layer is added, which reduces the spatial size of the data by selecting the maximum value from a local pool of pixels. This helps to reduce the number of parameters in the model and make it less prone to overfitting.Following the pooling layer, the data is flattened into a one-dimensional array and passed through a Dropout layer.

Dropout is a regularization technique that randomly drops out a fraction of the activations during training, helping to prevent overfitting by forcing the model to learn more robust features.Finally, a Dense layer with a softmax activation function is added for classification purposes. This layer outputs a probability distribution over the two possible classes in the model.

```
# metrics = [keras.metrics.AUC()]
metrics = ['accuracy']
opt = optimizers.SGD(learning_rate=0.01, momentum=0.9)
loss = 'binary_crossentropy'
# Compile model
model.compile(optimizer=opt,loss=loss, metrics=metrics)
```

```
batchsize = 25
epochs = 150
# Create data generators
train_datagen = ImageDataGenerator()
test_datagen = ImageDataGenerator()

train_generator = train_datagen.flow(X_train, y_train, batch_size=batchsize)
test_generator = test_datagen.flow(X_test, y_test, batch_size=batchsize)

# Train model
history = model.fit(train_generator,
                    steps_per_epoch=len(X_train) // batchsize,
                    epochs=epochs,
                    validation_data=test_generator,
                    validation_steps=len(X_test) // batchsize)
```

Snippet 7: Defining the metrics and the optimizing algorithm (TOP), and setting up the batchsize and the epochs of the model and fitting the model (BOTTOM).

Due to time and computation constraints of training such a model, a few hyperparameters were decided on after a few experiments.The Batch size and the number of epochs are two important hyperparameters used in training Convolutional Neural Networks (CNNs).Batch size refers to the number of training samples processed in a single forward/backward pass during training. In other words, the training data is divided into multiple batches, and the model is trained on each batch sequentially.

The batch size determines the amount of memory required to train the model and can also affect the accuracy of the model. Larger batch sizes can provide faster training times, but may cause the model to generalize poorly on the validation data.On the other hand, the number of epochs refers to the number of times the entire training dataset is passed through the network during training. Increasing the number of epochs allows the model to learn from the data more times, which can improve its accuracy. However, training for too many epochs can lead to overfitting, where the model memorizes the training data instead of learning the underlying patterns. Choosing the optimal batch size and number of epochs is a crucial step in training CNNs, as it can have a significant impact on the model's performance. It is important to strike a balance between the two hyperparameters, as the batch size and number of epochs can have an inverse relationship with each other. In general, a smaller batch size requires more epochs to achieve the same level of accuracy as a larger batch size. The model in Snippet 7 compiles a Convolutional Neural Network (CNN) model using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01 and momentum of 0.9 [13].

SGD is an iterative optimization algorithm that aims to minimize the loss function of a machine learning model. It works by taking small steps in the direction of the negative gradient of the loss function with respect to the model parameters. This process is repeated many times until the model reaches a minimum point in the loss function space.In image classification tasks, the goal is to train a model that can accurately classify images into their respective categories. The loss function used in these tasks is typically a categorical cross-entropy loss, which measures the difference between the predicted probabilities and the actual labels of the images.There are many optimization algorithms available for training deep neural networks in python, such as Adam, Adagrad, and RMSprop. However, SGD is a

popular choice due to its simplicity and effectiveness in many scenarios. SGD is computationally efficient, requires little memory, and has been shown to work well in practice for many image classification tasks. One advantage of SGD over other optimization algorithms like Adam is that it can help prevent overfitting, which occurs when a model becomes too complex and memorizes the training data instead of generalizing to new data. SGD achieves this by updating the model parameters based on a randomly selected subset of the training data, rather than the full dataset used by Adam. This random sampling of the data helps prevent the model from memorizing the training data and encourages it to learn more generalized patterns.

We mentioned that SGD minimizes a certain loss function, in this paper we propose using the binary cross-entropy loss function. In this loss function, the predicted output values are compared to the true output values, and the difference between them is quantified as the cross-entropy loss. The goal of the optimization algorithm, in this case, SGD, is to minimize this loss function to improve the model's performance.One advantage of using binary cross-entropy loss over other loss functions is that it is a simple and effective way to measure the difference between the predicted and true output values. The loss function is mathematically well-defined and has desirable properties that make it easy to optimize using algorithms like SGD. Another advantage of binary cross-entropy loss is that it is well-suited for probabilistic output models, such as logistic regression and neural networks, which are commonly used in image classification tasks. The loss function allows the model to output probabilities, which can be thresholded to make binary classifications.Finally, the compile method is called on the CNN model object, with the optimizer, loss, and metrics parameters specified. This method in Snippet 7 configures the model for training by specifying the optimizer to use, the loss function to minimize during training, and the evaluation metric to monitor during training.

```
test_loss, test_acc = model.evaluate(X_test, y_test)
```

```
6/6 [==============================] - 0s 4ms/step - loss: 1.3769 - accuracy: 0.7486
```

```
y_pred = model.predict(X_test)

# Compute confusion matrix
y_pred_classes = np.argmax(y_pred, axis=1)
conf_mat = confusion_matrix(y_test[:, 1], y_pred_classes)

# Get sensitivity and specificity
tn, fp, fn, tp = conf_mat.ravel()
se = tp / (tp + fn)
sp = tn / (tn + fp)

print('Sensitivity:', se)
print('Specificity:', sp)
```

```
Sensitivity: 0.3
Specificity: 0.8389261744966443
```

Snippet 8: Calculating both the accuracy (TOP) and the confusion matrix (BOTTOM) of the model.

After training the model using a portion of the available data, the remaining testing data was used to evaluate the performance of the model. The accuracy and confusion matrix were calculated to assess the model's ability to correctly classify the testing data. The results indicate that the model achieved an accuracy of 75%, with a sensitivity of 0.3 and specificity of 0.84. These findings demonstrate that the model's performance is comparable to the clinical benchmark, which has a sensitivity of 0.49 and specificity of 0.78 [20]. Furthermore, the potential for further exploration and modifications, including fine-tuning, presents an opportunity for improved results. Overall, this approach shows promise in achieving effective classification of the tested data.

Cloud Architecture

Networking

The first step of implementing the infrastructure for our CAD system is to set up the networking components so our system can communicate with each other. Multiple cloud providers provide multiple abstractions for networking. One of the main components of our

infrastructure is the VPC.A Virtual Private Cloud (VPC) is a secure and isolated virtual network within a cloud environment that allows users to deploy their resources in a private, customizable space. VPCs offer a high level of control and flexibility over network configuration, enabling users to define their own IP address ranges, subnets, and routing tables. This way, organizations can create their own virtual data center in the cloud, which can be accessed securely from anywhere, making it easier to manage and scale their resources. The security of VPCs is reinforced through advanced features like network access control lists, security groups, and VPN connections, providing additional layers of protection against cyber attacks.

It is really important to identify how to architect our network, there are multiple design patterns and the one we are proposing in this paper is the hub-spoke design pattern. Hub and spoke architecture is a popular design pattern that is widely used to create a scalable and flexible network infrastructure in cloud computing environments. This architecture involves the centralization of core resources, referred to as the hub, while distributing the remaining resources or spokes to peripheral locations. In this way, the hub acts as a central point of control, and the spokes serve as satellite locations where users and applications can access the services provided by the hub. This approach simplifies network management, improves security, and enhances overall performance by reducing the number of connections that need to be managed.

When it comes to implementing machine learning models and monitoring them, the hub and spoke architecture can be a valuable tool. By centralizing machine learning resources in the hub, such as the data lake, model training tools, and monitoring tools, organizations can ensure consistency and reliability in their machine learning workflows. The spokes can be used to deploy the machine learning models to the edge, enabling users to access the models from any location, while ensuring that the data stays secure and is compliant with relevant regulations.

Furthermore, the hub and spoke architecture can improve the monitoring and management of machine learning models by providing a central location for data collection and analysis. Monitoring tools can be deployed in the hub, allowing for the collection of data on model

performance, data quality, and other metrics. This data can then be analyzed to identify trends, anomalies, and other insights, allowing organizations to optimize their machine learning models and workflows. In summary, implementing a hub and spoke architecture in AWS can provide a scalable, secure, and efficient infrastructure for deploying and monitoring machine learning models.

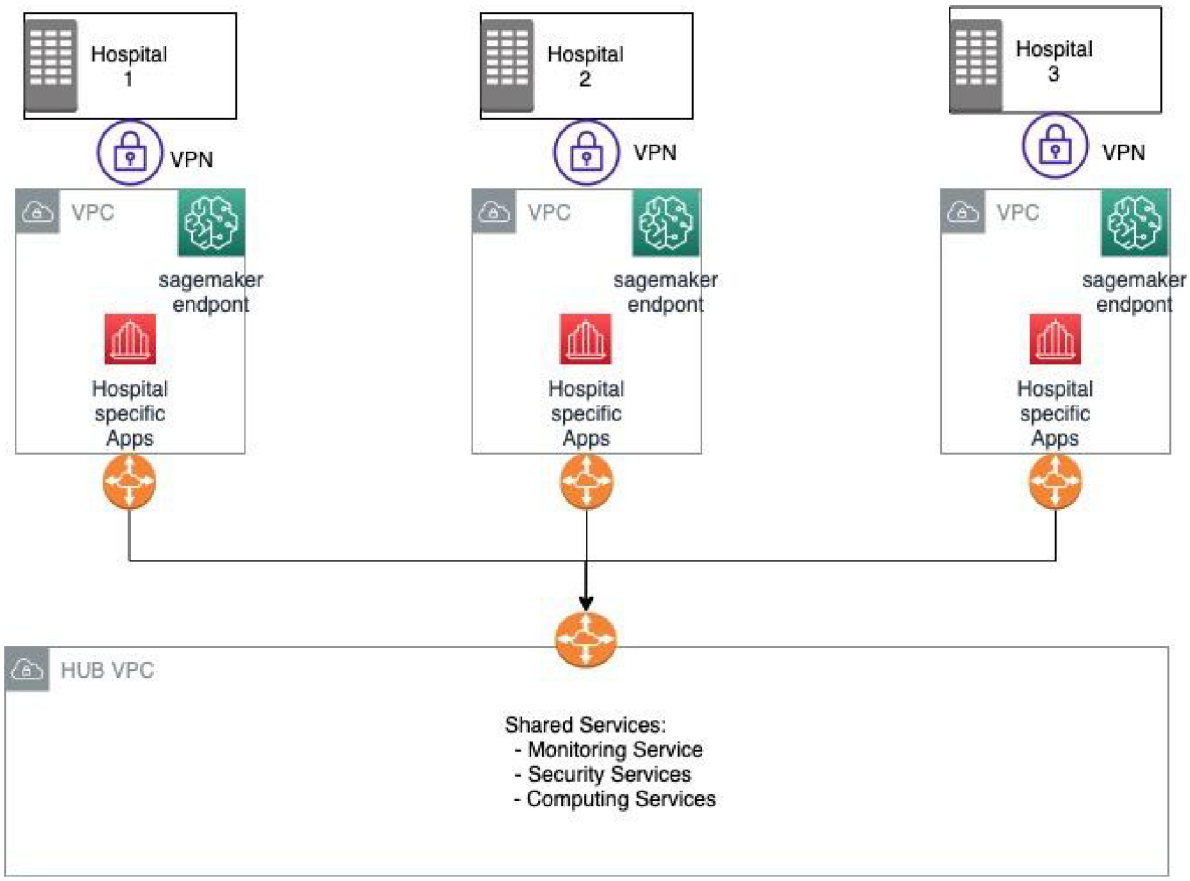


Figure 10: Proposed network architecture of the CAD system in which multiple hospitals could benefit from hosting the model on the cloud

As seen in Figure 10, the hub and spoke architecture can help us decouple our system components on the cloud in which we specify a VPC for each hospital that wishes to use the CAD system. Hospitals will be able to use the model to classify the FHR signals while giving the freedom to their R&D departments to design specific applications for their needs without affecting other hospitals or departments. In Figure 10, each hospital on premise systems will

be integrated to the cloud using a VPN, this ensures that sensitive data related to the patients are sent in a secure and encrypted way. Each VPC assigned to a hospital will have a sagemaker endpoint, we later in this paper introduce sagemaker and its capabilities but to put it simply, the sagemaker endpoint is just an api endpoint that will call our model to predict given an input. Finally, all VPCs will be connected to the main HUB VPC which allows us to monitor and secure the entire architecture using AWS managed services or the hospitals' IT services.

Data Storage

In this section, we introduce the CAD system and its various components, discussing how each component can be implemented. To illustrate the proposed solutions, we will use AWS as our cloud provider, though it is worth noting that most cloud providers offer similar services.

The first part of our system is the data storage. RDS or Amazon relational database service is proposed here. RDS on AWS offers numerous advantages over the storage server used in OB TraceVue. One of the primary benefits is that RDS is a fully managed database service, meaning that AWS takes care of much of the underlying infrastructure, such as server maintenance, backups, and software updates. This allows healthcare providers to focus on patient care rather than IT management. Additionally, RDS supports a range of relational databases, including MySQL, PostgreSQL, and Oracle, providing greater flexibility in terms of data storage options. Another key advantage of RDS is its scalability. With RDS, healthcare providers can easily scale their databases up or down as needed, depending on changing patient volumes or other factors. Finally, RDS provides strong security features, including encryption at rest and in transit, network isolation, and role-based access control. Setting up RDS is as simple as clicking a few buttons on the AWS portal. There are important things to keep in mind while creating an RDS for FHR signals like the database engine and the size of the database. For FHR signals, a suitable choice could be Amazon Aurora or MySQL, as they are both well-suited for storing large amounts of time-series data and can scale easily to accommodate growth in the dataset. Additionally, they both support high write volumes and provide robust data consistency and durability guarantees. Amazon Aurora, in particular, is a highly available and durable database engine that is designed for high-performance workloads.

It uses a distributed storage architecture that provides automatic failover and supports up to 15 read replicas, making it a good choice for applications that require high availability and low latency. It also supports storage autoscaling, which automatically increases storage capacity as needed, making it well-suited for use cases where the size of the FHR signals data may be unpredictable. The database will be responsible for storing the patient data as well as the FHR signals related to the patient.

In conventional FHR monitoring systems such as OB TraceVue, the FHR signal is typically acquired and stored in a dedicated database located on an internal server. To utilize the cloud-based RDS effectively, it is necessary to enable the real-time transfer of the FHR signal from the internal server to the cloud RDS, this could be simply made using Amazon Kinesis. Amazon Aurora is used to store the FHR signal data, however Amazon Kinesis will be used for streaming the data from OB TraceVue to Aurora. Kinesis can also help with real-time processing of the data as it's being streamed, allowing us to run analysis and machine learning models in real-time. Additionally, Kinesis allows for easy integration with other AWS services, such as Lambda functions or Amazon SageMaker for model training and deployment.

Amazon Kinesis can be used to ingest data from various sources, including FHR signals obtained from multiple servers in a medical environment. Kinesis data streams are used to collect and store this data in shards, which are the basic units of stream throughput, and enable parallel data processing. Kinesis client applications can be developed to consume the data from these streams and process it using AWS Lambda, Amazon Kinesis Client Library (KCL), or other custom applications. Kinesis also provides built-in integrations with other AWS services such as Amazon S3, Amazon Redshift, and AWS Lambda for easy processing, transformation, and storage of the streaming data. With Kinesis, healthcare professionals can easily and efficiently collect and process FHR data in real-time, enabling them to make informed decisions about patient care or to prepare them for modeling.

First, we would need to create a Kinesis data stream in AWS and configure it to receive data from the OB TraceVue system. This could involve setting up a Kinesis agent on the OB

TraceVue network to send the data to the Kinesis stream. Once the data is streaming into the Kinesis data stream, we could use AWS Lambda functions to process and transform the data as needed, and then load it into an RDS database in AWS. The Lambda functions can be triggered by events in the Kinesis data stream, such as the arrival of new FHR data. To ensure data security and compliance, we would also need to consider the encryption and access controls for the data both during transmission and storage. AWS provides various security features, such as encryption at rest and in transit, IAM policies, and VPCs, to help secure your data.

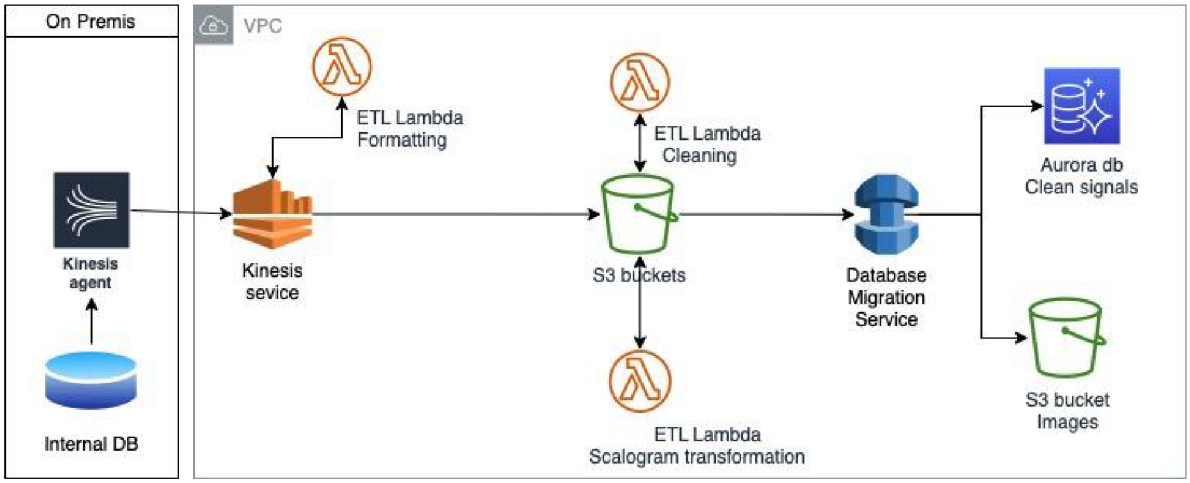


Figure 11: Infrastructure diagram to show the data streaming components proposed, which is responsible for streaming the FHR signals using Kinesis, transforming it, and loading to multiple S3 buckets to prep it for the model.

As seen in Figure 11, the Kinesis agent is installed on the on premise network and connected to the internal database where all the FHR signals are stored. The Kinesis service is responsible for streaming the data from on premise to the cloud. Different companies use different technologies to obtain the FHR signals and saves them into records in proprietary format, thus it is essential to format the signals into the needed format so we can use the fact that Kinesis service could be integrated with Lambda function and invoke a Lambda function to format the FHR signals. The S3 bucket will serve as a placeholder for all raw FHR signals in which we perform more ETL processes. The second Lambda function will be responsible

for cleaning the data, the function will be triggered as soon as the data is streamed to the S3 bucket and it will be coded in a similar way to Snippet 1.

The Lambda function will encapsulate the preprocessing steps described before and could be scaled when needed. The output of this function will be loaded into an S3 bucket. This S3 bucket will contain the clean FHR data which will serve as a single, authoritative source of information that can serve as a foundation for any future efforts to build a more dependable system. Researchers and doctors could now easily access those clean FHR data to expand their research without worrying about collecting the data. In this scenario, all FHR signals will be easily accessible and available.

We then use the AWS database migration which is a fully managed service, that allows us to easily migrate data from one source to another, in our case, the migration of denoised FHR signals takes place from the S3 buckets to the aurora database and to other S3 buckets specifically for images storage.

Data Modeling and Computing

MLOps is an emerging discipline that seeks to apply DevOps practices to machine learning workflows. By automating the entire machine learning lifecycle, MLOps can help to reduce the time, cost, and complexity of building, training, and deploying machine learning models. One of the key benefits of MLOps is its ability to enable the scaling of machine learning models to meet the needs of modern enterprise applications. In particular, Convolutional Neural Networks (CNNs) have been widely used for image classification tasks, leveraging the power of deep learning algorithms to identify and classify images. To leverage the benefits of CNNs on the cloud, AWS SageMaker can be used to train, test, and deploy the model to classify images obtained from scalogram of FHR signals stored in the S3 bucket.

AWS SageMaker is a fully-managed service that provides a comprehensive platform for building, training, and deploying machine learning models at scale. With SageMaker, users can quickly and easily build and train machine learning models using a variety of built-in

algorithms or custom code. The service automates the process of training and tuning models, enabling developers and researchers to focus on developing the best algorithms for their specific use case. SageMaker also includes capabilities for deploying and managing machine learning models in a scalable and cost-effective manner, allowing organizations to easily integrate machine learning into their applications.

By using AWS SageMaker to train, test, and deploy the CNN model to classify images obtained from scalogram of FHR signals stored in an S3 bucket, organizations can realize several benefits. Firstly, the cloud-based approach enables the processing of large datasets in a scalable and cost-effective manner.Secondly, SageMaker simplifies the development, training, and deployment of machine learning models, enabling organizations to rapidly iterate and optimize their models for accuracy and performance. Finally, by leveraging AWS storage services such as S3, organizations and clinics can benefit from highly available, secure, and durable end to end solutions, ensuring that the service is always accessible, protected and reliable.

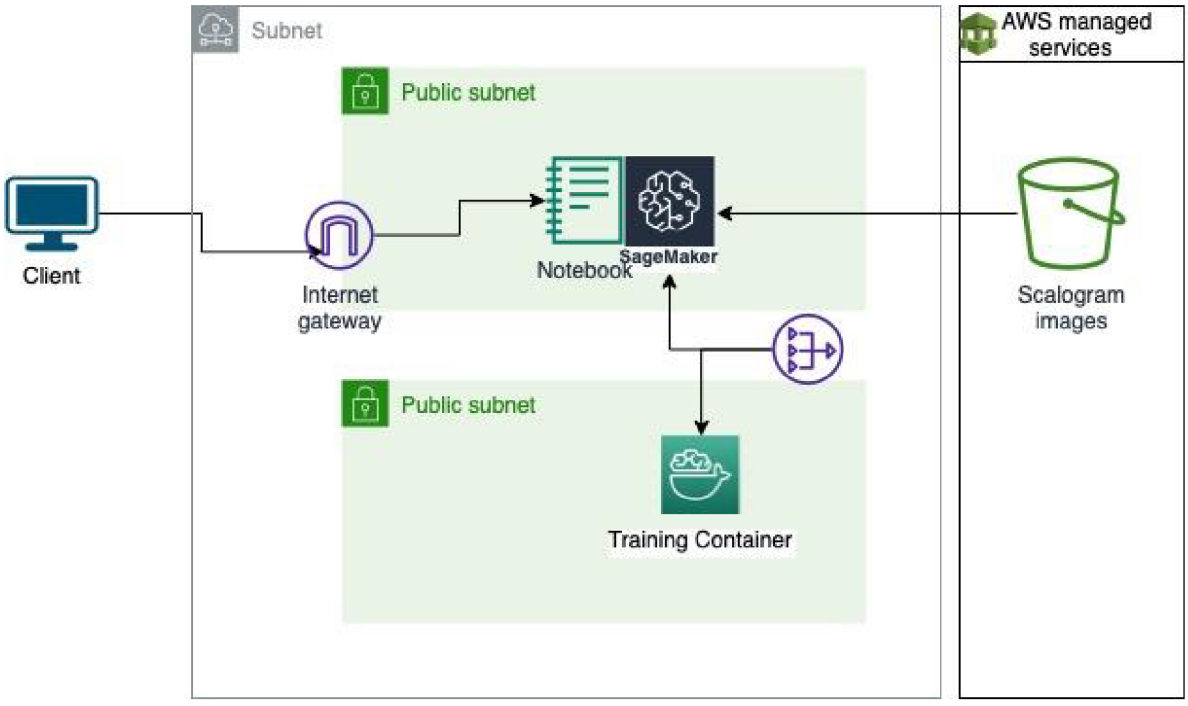


Figure 12: Showing the MLOPS architecture of the model deployed on AWS, a subnet containing an instance of the machine learning model, while another subnet contains computing containers that scale to train the model. This subnet will be placed in the hub VPC

Security and Compliance

The use of MLOps and AWS SageMaker for image classification tasks has significant implications for security and compliance. With the increasing importance of data privacy and security, organizations need to ensure that their machine learning models do not compromise sensitive information. By building, training, and deploying machine learning models on the cloud, organizations can leverage the security and compliance features of AWS, including encryption, access control, and audit trails.

AWS provides a range of security features that are designed to help customers meet their compliance requirements. These include robust identity and access management capabilities, network security controls, and data protection features. AWS also provides compliance programs, such as PCI DSS, HIPAA, and SOC 2, which enable customers to meet their regulatory obligations. Additionally, AWS offers a range of services that can help customers manage their security and compliance requirements, such as AWS Config, AWS CloudTrail, and AWS Security Hub.

With MLOps and SageMaker, organizations can ensure that their machine learning models are built, trained, and deployed in a secure and compliant manner. By leveraging the security and compliance features of AWS, organizations can build and deploy models that are protected from unauthorized access and ensure that data privacy is maintained. Additionally, the use of MLOps practices enables organizations to manage the entire machine learning lifecycle, including testing, deployment, and monitoring, in a secure and compliant manner.

V. Results and Discussion

In the modeling part in section IV, we fitted a CNN model on the training images created from the scalogram and we were able to obtain relatively satisfactory results close to the clinical benchmark with specificity even surpassing the benchmark, however we believe that there is room for improvements. The first observation of our model is that the model is struggling to identify the pathological signals thus the low sensitivity. One possible reason for the low sensitivity could be the imbalanced dataset. If the number of normal cases in the dataset is much higher than the number of pathological cases, the model may not be able to learn enough information about the pathological cases, leading to poor performance on these cases. In such cases, using data augmentation techniques or oversampling the pathological cases may help to balance the dataset and improve the model's sensitivity.

One common technique is to duplicate existing samples from the minority class to match the number of samples in the majority class. However, this may lead to overfitting, where the model learns to simply memorize the duplicated samples. Another approach is to generate synthetic samples using techniques such as Synthetic Minority Over-sampling Technique (SMOTE), where new samples are created by interpolating between existing samples in the minority class. This can help to provide new information to the model and improve its performance on the minority class.

Data augmentation techniques can also help to improve model performance by generating new training samples from existing ones. For image data, this can include techniques such as rotation, scaling, and flipping. In the context of scalogram images, additional techniques such as adding noise, blurring, or changing the color map could be used. These techniques can help to simulate variations in the data and provide the model with more diverse examples to learn from. We propose for future implementation to include one of those techniques to help increase the sensitivity of the model.

Another novel and creative technique used to treat imbalanced data is creating Generative Adversarial Networks as described in [21]. Generative Adversarial Networks (GANs) are a

type of deep learning algorithm that involves two neural networks, a generator and a discriminator, that work together to generate new data samples that are similar to a given dataset.

The generator network learns to generate synthetic data samples that are similar to the real data, while the discriminator network learns to distinguish between the real and synthetic samples. The two networks are trained in a feedback loop, where the generator tries to produce more realistic samples to fool the discriminator, while the discriminator tries to identify the synthetic samples.

GANs can be used to treat imbalanced data by generating synthetic data samples for the underrepresented class, thus increasing the size of the minority class and balancing the dataset. This can improve the performance of classification models by providing more training examples for the minority class, which can be important in many real-world applications where the cost of misclassification is high for the minority class. However, it's important to note that the synthetic samples generated by GANs may not always accurately reflect the underlying distribution of the data, so their usefulness for imbalanced data depends on the specific application and the quality of the generated samples. *Puspitasari, Riskyana Dewi, et al* [21] were able to showcase this approach in their paper and successfully were able to increase the performance of multiple different models.

Another reason could be the feature engineering and by feature engineering here we imply all the steps taken to produce the training data for our model, including the preprocessing steps. One vital observation worth mentioning is that even after denoising the signal, we might have not cleaned the signal entirely, as removing the artifacts could have introduced new artifacts, some are even visible in Figure 8. We can see in that figure that we removed most of the noise, however we notice visible spikes in the FHR which might have hindered the CNN model performance, so we suggest also interpolating the values in the FHR signals that causes this spikes and after careful reviewing we found out that [13] also mentioned this problem in his paper and how it is treated.

In addition to that we believe the scalogram transformation might not be enough. While the scalogram images may provide useful features for classification, it is possible that there are other features in the data that could help to improve the model's performance. For example, extracting time-domain features such as mean and standard deviation of the signal may provide additional information that the model can use to improve its sensitivity. Since this paper focused entirely on letting the model choose its own features without implicitly extracting the important features, this idea was discarded, however some publications were able to benefit from both approaches by creating an ensemble model that combines both machine learning model fed with time domain extracted features and a deep learning model.

The choice of wavelet function for generating the scalogram can also affect the sensitivity of the CNN model. In our case, we used the Morlet wavelet to generate the scalogram images from the FHR signals. However, different wavelet functions may have different frequency responses and resolutions, which can impact the representation of the signal in the image and, in turn, affect the performance of the model. For example, if the Morlet function has poor resolution at the frequency range where the pathological signals are concentrated, the resulting scalogram images may not contain enough information to enable the model to accurately classify these signals. In contrast, choosing a wavelet function that has better resolution in the relevant frequency range may result in more informative images, and potentially lead to higher sensitivity. So experimenting with different wavelets could lead to better sensitivity however the choice of the Morlet function was taken due to the reasons described in section III.

Finally, the model architecture may also play a role in the low sensitivity. Different architectures have different strengths and weaknesses. Experimenting with different architectures, such as adding more layers or changing the activation functions, may help to improve the model's sensitivity.

VI. Conclusion

To conclude this paper, we start by summarizing the problem that obstetricians face. Obstetricians face the challenge of accurately interpreting FHR signals and identifying any

abnormalities that may indicate hypoxia, a condition where the fetus is not receiving sufficient oxygen supply. It is critical to analyze FHR signals promptly and interpret them correctly to ensure the well-being of both the fetus and the mother. By detecting abnormalities early, doctors can make informed decisions regarding the delivery process, potentially reducing the number of unnecessary cesarean operations resulting from misinterpretation of FHR signals and false concerns about fetal health. The accurate and timely analysis of FHR signals is crucial in ensuring a safe and healthy delivery process.

We then proposed a CAD system that utilizes deep learning to help doctors classify and analyze FHR signals and discussed the challenges and benefits of creating such a system. We explained the main components of the system which includes the data, the model and the infrastructure

Each component was examined starting with the data in which we discussed multiple preprocessing steps that are necessary for signal denoising and cleaning and then explained the different techniques researchers used to transform FHR signals into more meaningful representations. We explored wavelet transformations and pointed out its advantages as well as its limitations and proceeded with transforming the signals into 2D images. Those images were then fed to a CNN model to be trained and then tested the model on a completely different subset of the data than the one used for training to classify the images into normal and abnormal. Even though the specificity of the model (84%) was higher than the clinical benchmark, we observed low sensitivity (30%) which made us investigate more on why that happened and concluded with multiple reasoning behind that, in which the occurrence of imbalance data was the main culprit.

We finally proposed the framework of deploying the model to the cloud for it to be used and integrated with the existing infrastructure. We explained why deploying to the cloud will be beneficial and can help us overcome a lot of the challenges that can occur from deploying an AI system to an existing environment. We mentioned the security benefits as well as the scalability, availability and reliability of such systems.

In the end, this paper is aimed to reduce the gap between academic research and real-world healthcare settings while showcasing the advantages of closing this gap. By demonstrating the practical benefits of such a system, this paper aims to accelerate the integration of AI solutions in medical care and highlight the potential for these technologies to enhance patient outcomes. Ultimately, the hope is that this work can help facilitate a more seamless integration of AI in medical settings and advance the field of obstetrics by improving the accuracy and efficiency of FHR signal analysis.

VII. References

[1] Chudáček, V., et al. "Automatic Evaluation of Intrapartum Fetal Heart Rate Recordings: A Comprehensive Analysis of Useful Features." Physiological Measurement, vol. 32, no. 8, July 2011, pp. 1347–60, https://doi.org/10.1088/0967-3334/32/8/022. Accessed 30 Mar. 2023.

[2] Lavender, Tina, et al. "Effect of Different Partogram Action Lines on Birth Outcomes." Obstetrics & Gynecology, vol. 108, no. 2, Aug. 2006, pp. 295–302, https://doi.org/10.1097/01.aog.0000226862.78768.5c. Accessed 3 May 2022.

[3]Alfirevic, Zarko, et al. "Continuous Cardiotocography (CTG) as a Form of Electronic Fetal Monitoring (EFM) for Fetal Assessment during Labour." Cochrane Database of Systematic Reviews, vol. 2, no. 2, 3 Feb. 2017,

[4] Blix, Ellen, et al. "Inter-Observer Variation in Assessment of 845 Labor Admission Tests: Comparison between Midwives and Obstetricians in the Clinical Setting and Two Experts." Obstetrical & Gynecological Survey, vol. 58, no. 6, June 2003, pp. 371–373, https://doi.org/10.1097/01.ogx.0000070123.68903.1e. Accessed 13 Dec. 2021.

[5] Ayres-de-Campos, D., Spong, C. Y., Chandraharan, E., & FIGO Intrapartum Fetal Monitoring Expert Consensus Panel. (2015). FIGO consensus guidelines on intrapartum fetal monitoring: cardiotocography. International Journal of Gynecology & Obstetrics, 131(1), 13-24.

[6] Topol, Eric J. "High-Performance Medicine: The Convergence of Human and Artificial Intelligence." Nature Medicine, vol. 25, no. 1, Jan. 2019, pp. 44–56, https://doi.org/10.1038/s41591-018-0300-7.

[7] Kumar, Pranjal, et al. "Artificial Intelligence in Healthcare: Review, Ethics, Trust Challenges & Future Research Directions." Engineering Applications of Artificial Intelligence, vol. 120, Apr. 2023, p. 105894, https://doi.org/10.1016/j.engappai.2023.105894. Accessed 30 Jan. 2023.

[8] Chudáček, Václav, et al. "Open Access Intrapartum CTG Database." BMC Pregnancy and Childbirth, vol. 14, no. 1, 13 Jan. 2014, https://doi.org/10.1186/1471-2393-14-16. Accessed 1 Apr. 2020.

[9] Ramanah, Rajeev, et al. "Predicting Umbilical Artery PH during Labour: Development and Validation of a Nomogram Using Fetal Heart Rate Patterns." European Journal of Obstetrics & Gynecology and Reproductive Biology, vol. 225, June 2018, pp. 166–171, https://doi.org/10.1016/j.ejogrb.2018.04.008. Accessed 30 Mar. 2023.

[10] Kashanian, M., et al. "Pregnancy Outcome Following a Previous Spontaneous Abortion (Miscarriage)." Gynecologic and Obstetric Investigation, vol. 61, no. 3, 2006, pp. 167–170, https://doi.org/10.1159/000091074. Accessed 12 Apr. 2021.

[11] Goudar, Shivaprasad S., et al. "Stillbirth and Newborn Mortality in India after Helping Babies Breathe Training." Pediatrics, vol. 131, no. 2, 1 Feb. 2013, pp. e344–e352, https://doi.org/10.1542/peds.2012-2112. Accessed 31 Mar. 2023.

[12] Zhao, Zhidong, et al. "A Comprehensive Feature Analysis of the Fetal Heart Rate Signal for the Intelligent Assessment of Fetal State." Journal of Clinical Medicine, vol. 7, no. 8, 20 Aug. 2018, p. 223, www.ncbi.nlm.nih.gov/pmc/articles/PMC6111566/, https://doi.org/10.3390/jcm7080223. Accessed 8 Oct. 2021.

[13] Zhao, Zhidong, et al. "DeepFHR: Intelligent Prediction of Fetal Acidemia Using Fetal Heart Rate Signals Based on Convolutional Neural Network." BMC Medical Informatics and Decision Making, vol. 19, no. 1, Dec. 2019, https://doi.org/10.1186/s12911-019-1007-5. Accessed 31 Mar. 2023.

[14] Casas, O., et al. "Fetal acidemia prediction using fetal heart rate signal analysis and support vector machine." IEEE Transactions on Biomedical Engineering, vol. 65, no. 3, 2018, pp. 521-530.

[15] Zhu, Mengni, and Liping Liu. "Fetal Heart Rate Extraction Based on Wavelet Transform to Prevent Fetal Distress in Utero." Journal of Healthcare Engineering, vol. 2021, 29 Sept. 2021, pp. 1–7, https://doi.org/10.1155/2021/7608785. Accessed 5 Feb. 2023.

[16] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(1), 1929-1958

[17] Zhao, Y., Gao, X., Li, M., Wang, S., & Zhang, D. (2019). DeepFHR: intelligent prediction of fetal acidemia using fetal heart rate signals based on convolutional neural network. BMC Medical Informatics and Decision Making, 19(Suppl 7), 270.

[18] Ozturk, Ibrahim, et al. "Fetal Heart Rate Classification using Deep Neural Networks and Wavelet Transform." Expert Systems with Applications, vol. 160, 2020, p. 113698. ScienceDirect

[19] Ponsiglione, Alfonso Maria, et al. "A Comprehensive Review of Techniques for Processing and Analyzing Fetal Heart Rate Signals." BioMed Research International, vol. 2017, 2017, pp. 1-15, doi:10.1155/2017/7906284.

[20] Abry, P., Spilka, J., Leonarduzzi, R., Chudáček, V., Pustelnik, N., & Doret, M. (2018). Sparse learning for intrapartum fetal heart rate analysis. Physiological Measurement, 39(5), 054002. doi: 10.1088/1361-6579/aababb

[21] Puspitasari, Riskyana Dewi, et al. "Generative Adversarial Networks for Unbalanced Fetal Heart Rate Signal Classification." ICT Express, vol. 8, no. 2, 2022, pp. 239–243., https://doi.org/10.1016/j.icte.2021.06.007.