POLITECNICO DI MILANO

School of Industrial and Information Engineering

Master of Science in Computer Science and Engineering

Artificial Intelligence techniques for the classification of Cardiotocographic signals

Advisor:

Prof. Signorini Maria Gabriella

Co-Advisor:

Dott. Ing. Edoardo Spairani

Author:

Beniamino Daniele

905347

Academic Year 2020/2021

Abstract

Cardiotocography is one of the most used tool to clinically evaluate the well-being of the fetus. This thesis presents a novel machine learning approach applied to classifying disease states in fetuses during pregnancies. The study has evaluated CardioTocoGraphic (CTG) traces that provide Fetal Heart Rate signal and Uterine contractions. Other features are the mother's age and the gestational week. Due to the increasing success of machine learning models in the classification environment, this work explores both machine learning and deep learning models to understand which performs better. Multiple machine learning algorithms have been studied while exploiting features selection techniques to have various subsets on which to test the models. Instead, our deep learning models focus not only on learning from the above-said data but also on images extracted from CTG signals utilizing mathematical models that allow signal-to-image transformations. By exploiting multiple architectures like Multi-Layer Perceptron, Convolutional Neural Networks, and Long Short-Term Memory, we use a hybrid approach by combining these models. Our results show that our deep learning models perform better than machine learning models by reaching an 80% accuracy compared to 70%. This opens a possible integration in the clinical environment. The proposed approach could integrate the existing processing techniques on the evaluation of fetal disease states.

Introduzione

La Cardiotocografia è uno degli strumenti più utilizzati per analizzare lo stato di benessere del feto. Questa tesi presenta un approccio di machine learning per la classificazione di stati patologici del feto nella gravidanza. Lo studio si è basato sui traccati della CTG contenenti il segnale del battito fetale e le contrazioni uterine. Altre informazioni usate sono l'età materna e la settimana gestazionale. Grazie al continuo successo del machine learning nel classificare set di dati, questo lavoro esplora tecniche di machine learning e deep learning per confrontare e capire quale modello possa performare meglio. Nello specifico multipli algoritmi di machine learning sono stati usati, sfruttando tecniche di features selection per creare più sottoinsiemi di dati su cui poter testare i modelli. Per i modelli di deep learning sono stati usati non soltanto i dati sovramenzionati, ma anche immagini estratte dai segnali provenienti dalla CTG usando modelli matematici che permettono la conversione di un segnale in immagine. Avendo usato più architetture come il Multi Layer Perceptron, Convolutional Neural Networks e Long Short-Term Memory, abbiamo pensato di usare un approccio misto combinando questi modelli. Questa tesi mostra come i modelli di deep learning performano meglio rispetto a quelli di machine learning raggiungendo un'accuratezza

dell'80% rispetto ad un 70%. Questi risultati portano ad una possibile integrazione nell'ambiente clinico. L'approccio proposto potrebbe integrare le già presenti tecniche di processamento sulla valutazione degli stati patologici del feto.

Extended Abstract

Cardiotocograph traces are widely used since their interpretation represents the golden standard for monitoring the well-being of the fetus. However there is a lack of guidelines regarding the visual inspection and interpretation of the traces that brings doctors in having discordant interpretations. In fact, complex signal changes in the short period are almost undetectable. For this reason, a combination of machine learning and deep learning approaches could help in clinical decisions.

In this thesis, we based our work on a software named 2CTG. A CTG trace contains two signals: fetal heart rate (FHR) and uterine contractions. These signals are sent from the cardiotocograph to a laptop, that analyzes all the information recorded. These data are then sent to another laptop that contains the software 2CTG that examine the data based on linear and non-linear parameters. The data are sampled with an interval of 500ms. After the sampling, signals are averaged over periods of 2.5s to avoid redundancy.

CTG recordings may contain artifacts due to two phenomenons: double-counting (the FHR is sampled two times and summed) half-counting (only half of the signals are sampled). The variability is an important parameter extracted from the FHR, as it can indicate a pathological status of the fetus, like severe hypoxia. It can be measured on a three-time scale: Short Term Variability (STV), medium-term (Delta), Long Term Irregularity (LTI). As the pregnancy progresses over time also the variability increases, but in pathological situations, the variability decreases creating possible and dangerous consecutive deceleration. This thesis aims to create a framework for the classification of CTG signals using machine learning and deep learning algorithms.

The dataset used in this work comes from the Hospital Federico II of Naples. It comprises 9476 pregnant women with 24095 cardiotocographic traces. The features chosen to be important are the date of birth, the health status of the patient (which was used to divide our dataset into healthy and pathological), the date in which the CTG trace was recorded, the gestational week, and other linear and non-linear parameters that can be extracted from the FHR signal like STV, LTI, DELTA, number of accelerations and decelerations. The signals, before being used by our algorithms, had to be cleaned of artifacts. A modified version of Mantel's algorithm was used to extract the baseline. We were able to extract the linear and non-linear features from the signal that enriched our dataset.

Lastly, we used multiple signal-to-image techniques to generate our last part of the dataset, crucial for the Convolutional Neural Networks, present in our work. The FHR signals were split into sub-sequence of 20 minutes each containing 2400 sampled data. Then they were transformed using Continuous Wavelet Transform (CWT), Gramian Angular Summation/Difference Field (GASF/GADF), Markov Transition Field (MTF), Recurrence Plot (RP), Self-Similarity Matrix (S), Power Spectrogram (PS), and Persistence Spec-

trum (PSP). With these transformations, we reached a total of 48398 images, of which 20588 were healthy and 27810 were pathological.

Regarding the machine learning approach, we applied feature selections techniques like Univariate Feature Selection (UFS), Recursive Feature Elimination (RFE), Decision Tree (DT), and Principal Component Analysis (PCA) to eliminate redundant or irrelevant variables. The features got scaled-down between the range (0,1) using the MinMax method. The models used are Logistic Regression (LR), Random Forest (RF), K-Nearest Neighbors (k-NN), Adaboost, Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Bagging, Gradient Boosting, Extreme Gradient Boosting (XGB).

The hyperparameters have been tuned using Randomized Search with 5-fold cross-validation. For the reference model "Logistic Regression" we used L2 regularization with C equal to 10, using liblinear and lbfgs as solvers. For the Support Vector Machine, we use C equal to 13 with rbf as kernel. For the Multi-Layer Perceptron, we used 4 hidden layers, respectively: 500, 250, 150, 50. For the k-NN, the neighbors were 5, with metric Manhattan distance. For the Bagging, the estimators chosen was the SVM, with 20 estimators. For the Gradient Boosting the estimators were 200 with a learning rate of 0.1. For AdaBoost, the estimators were 100. For the XGB we used 100 estimators with a learning rate of 0.1. For the Random Forest, we used 100 estimators with entropy as the criterion.

In our results we added values like Recall, F1-Score, R^2, Root Mean Squared Error and Area Under the Curve. The overall best performing model is XGB as it was the model with the highest R^2, for all the subsets generated by each feature selection technique. Using Univariate Selection the best performing models are Random Forest and XGB achieving a 0.7 R^2, RMSE 0.56, F1-Score 0.68, Recall 0.68, and an AUC of 0.682.

Since the results achieved with Machine Learning were not as we expected, we tested also the performances deep learning algorithms. To avoid overfitting we used regularization techniques like weight decay, batch normalization, and drop out. Our first model is an MLP where all the dense layers use the ReLU function except for the last layer which uses the softmax for classification. The dataset has been split for all the models between training and test (80%/20% respectively) and cross-validation has been performed. The layer of this model is composed as following: dense layer of 500 neurons, 40% dropout, dense layer of 250 neurons, 40% dropout, dense layer of 150 neurons with both L1 and L2 regularization penalty, bias and activity regularizers L2, dense layer of 50 neurons, 40% dropout and output layer 2 neurons. The value for L1 is $1e^{-5}$, for L2 is $1e^{-4}$. We have used as optimizer Adam with learning rate 10^{-4} and decay $\frac{10^{-4}}{200}$. The loss function used is the binary cross-entropy. With this model, the accuracy achieved is 75.7%.

Our next goal was to use the power of Convolutional Neural Networks by feeding them the images extracted using the following tools: Continuous Wavelets Transform, Gramian Angular Summation Field, Gramian Angular Difference Field, Markov Transition Field, Self-Similarity Matrix, Recurrence Plot, Power Spectrogram, and Persistence Spectrum. We extracted various images sizes 32x32x3, 64x64x3, 128x128x3 and 256x256x3 and tried multiple combinations of images to understand which were more meaningful for the model. To run this model we used the website Kaggle as it provided GPU computation that speeded up the training. Although the main limitation of using Kaggle was the maximum RAM available capped at 13GB, that did not allow us to train our model on images greater than 64x64x3, as it would have required more than 15GB of memory.

The CNN model configuration was inspired by the LeNet architecture. The convolutional layers use the ReLU function except for the last layer which uses the softmax function for classification. The model structure is as follow: Convolutional Layer with 16 filters and a 5x5 kernel, batch normalization layer, max pooling layer 2x2, convolutional layer with 32 filters and 5x5 kernel, batch normalization layer, max pooling layer 2x2, 80% dropout, flatten layer, dense layer with 64 neurons, batch normalization, dropout 80%, dense layer with 16 neurons and the final output layer with 2 neurons. The model reached a 68% accuracy with the 64x64x3 images, compared to 64% of the 32x32x3 images. These results are congruent to the fact that the model should be able to learn better in discriminating classes by having more information.

By having the raw FHR's signal, we decided to test Recurrent Neural Networks, that perform great in temporal predictions. Since RNNs suffer the problem of vanishing/exploding gradient, we opted for LSTMs. We used as dataset only the signals with 2400 data points, and if there were more, we resorted to an overlapping technique to create multiple signals. The dataset has been scaled down using the MinMax function. The model architecture is simple: LSTM layer with 8 units, batch normalization, LSTM layer with 8 units, batch normalization, LSTM layer with 8 units, and an output layer of 2 neurons with softmax as the activation function for classification. The accuracy reached is 55%, which indicates that our model was almost as good as random guessing. We think that this scarce result is due to the nature of the signal. In fact, the FHR is a stochastic signal compared to an almost periodic heartbeat signal, making it difficult for the LSTM in finding patterns inside the signal.

Lastly, we have tried multiple hybrid approaches that concatenate the last layers in the above architectures and attach an additional dense layers of 128 neurons and activation function ReLU before the final output layer. The best performing hybrid model is the MLP combined with the CNN, as it reached an averaged accuracy of 79.1% with a peak accuracy of 85.2% during the training session.

Another combination was the MLP with CNN and LSTM. This model reached an accuracy of 78.9%, slightly lower than the MLP with the CNN. Confirming that the LSTM is not providing additional information, but it is slightly penalizing the model. The last model is the MLP with the LSTM. It reached the least accuracy, as the LSTM did not add any information to the MLP. In fact, the accuracy remained the same as for the MLP 75.7%.

We believe that the accuracy of our models could be further increased by training our models on images bigger than 64x64x3 as they would add more information, since the limited resolution might reduce the ability of the model to learn and improve and by having more data about the mother's general status like BMI, blood pressure and so on.

Lastly, we think that the LSTM training was impaired by the stochastic nature of the signal, making it hard to find correlations. Especially because some signals have been corrected by interpolating some points. We want to end by adding some other further future improvements: use other temporal aspects of the signal in association to the FHR for the LSTM training, explore other images transformations, re-calculate the mean values for the signal and create a website with a user interface, that has our pre-trained deep learning model in the backend allowing doctors to upload their CTG traces and get a prediction of the state of the pregnancy.

Contents

1 Cardiotocography **1**

1.1 Pregnancies and Risks Factors 1

1.2 History of Cardiotocography 2

1.2.1 Fetal Heart Rate 4

1.2.2 Tocography 11

1.3 State of the Art 12

2 Database & Preprocessing **14**

2.1 Dataset 14

2.1.1 2CTG Software 14

2.1.2 Data extraction 17

2.1.3 Data Labeling 22

2.2 Pre-Processing 24

2.2.1 Signals correction and baseline extraction 24

2.2.2 Final dataset 26

2.3 Tools & libraries 27

3 Machine Learning **30**

3.1 CTG's Features 30
3.1.1 Time Domain Parameters 31
3.1.2 Frequency Domain Parameters 33
3.1.3 Non-linear Parameters 34
3.2 Feature Selection 36
3.2.1 Principal Component Analysis 36
3.3 Feature Scaling 41
3.4 Hyperparameters Tuning 41
3.5 Algorithms 41
3.6 Results 44
3.7 Limitations 45

4 Deep Learning **48**
4.1 Introduction to Neural Networks 48
4.1.1 Avoid Overfitting 51
4.2 Multilayer Perceptron 52
4.2.1 Our Model 54
4.2.2 Results Achieved 55
4.3 Convolutional Neural Networks 56
4.3.1 Images Transformation 56
4.3.2 Our Model 66
4.3.3 Results Achieved 68
4.4 Recurrent Neural Networks 69
4.4.1 Long Short-Term Memory 70
4.4.2 Our Model 72

4.4.3 Results Achieved 72
4.5 Mixed Approach 73
4.5.1 MLP & CNN 73
4.5.2 MLP & LSTM 74
4.5.3 MLP & CNN & LSTM 75
4.5.4 Final Results 76

5 Conclusions **79**
5.1 Machine Learning versus Deep Learning 79
5.2 Limitations 81
5.3 Future Work 82

List of Figures

2.1 2CTG User interface with a CTG tracing 15

2.2 Example of a signal containing artifacts 25

2.3 Example of a signal after the correction 26

3.1 Plotted dataset and its principal components 38

3.2 Explained variance by each feature 39

3.3 Cumulative explained variance 40

4.1 Example of a Multi Layer Perceptron 53

4.2 Example of a Continuous Wavelet Transformation image . . . 58

4.3 Example of a Gramian Angular Difference Field Image 59

4.4 Example of a Gramian Angular Summation Field Image . . . 60

4.5 Example of a Markov Transition Field Image 61

4.6 Example of a Self Similarity Matrix Image 62

4.7 Example of a Recurrence Plot Image 63

4.8 Example of a Power Spectrum Image 65

4.9 Example of a Persistence Spectrum Image 66

4.10 Our CNN model 68

4.11 LSTM architecture 70

5.1 Performance of Deep Learning and Machine Learning with respect to the amount of data 80

List of Tables

1.1 Guidelines 7

2.1 Records example in ANAGRAFE_PCC table 17

2.2 Records example in ANAGRAFE_PCC table 21

2.3 Records example in ESAME_CTG table 21

2.4 Records example in PARAMETRI_CTG table 21

3.1 Features' ID 37

3.2 Features Selected by each algorithm 46

3.3 Classifiers trained on datasets with different features 47

4.1 Top 10 best performing combinations of images for the CNN . 77

4.2 Top 10 best performing combinations of images for the mixed model (CNN+MLP) 77

4.3 Top 10 best performing combinations of images for the mixed model (CNN+MLP+LSTM) 78

4.4 Best results of the models tested 78

Chapter 1

Cardiotocography

This work aims to create a tool that can help doctors discerning if the fetal heart during pregnancy is normally developing or if pathological fetal changes are in progress. As it can be easily understood, the job is complex, and the objective is ambitious. To accomplish this goal, we used machine learning techniques and deep learning algorithms to analyze the temporal series of the CardioTocoGraphic(CTG) signals as well as numerical data related to the mother and signals. The temporal series have been transformed in images to exploit the power of *Convolutional Neural Networks* (CNN) in understanding peculiar details.

1.1 Pregnancies and Risks Factors

Although most pregnancies proceed physiologically, approximately 8% of them have complications. [1] These complications might arises due to adverse mother's health conditions, thus leading to various medical conditions further impacting the health of both the mother and fetus.

The negative impact on the fetus's health is usually referred to as "fetal distress", which is linked to an alteration and decrease in the Fetal Heart Rate (FHR).

1.2 History of Cardiotocography

Cardiotocography (CTG) was introduced to monitor the health status of the fetus and currently is the most used technique. A typical CTG pattern is considered a good indicator for the fetus's well-being thanks to its wide adoption and the amount of data gathered, but the opposite is not true.

The efficacy of the CTG antepartum decreases if there is a need to identify fetal distress. At the core of this problem, there is a lack of guidelines regarding the interpretation of the CTG. There are currently numerous visual reading systems, but still, no standard has been reached. Even if in the same center the same guidelines are followed, discordant interpretation might arise.

As a matter of fact, complex signal changes in the short period are almost impossible to detect with the naked eye. Hence, it is possible to conclude that the CTG visual reading is inaccurate and thus cannot extract all the information inside the signal, despite its large use. To process CTG signals we used a software named 2CTG, adopted by the Hospital Federico II of Naples. [2]

The CTG is comprised of two components: a cardiotocograph, which captures the FHR signal, uterine contractions, and fetal motor activity through

Doppler or heartbeats, and a software installed on an ordinary laptop that analyzes all the information captured by the former, then the trace is sent to another laptop containing the software 2CTG, which analyzes all the sample based on the linear and non-linear parameter of the FHR. [3]

The system 2CTG, that we used, reads:

1. time interval between two following cardiac contractions
2. activation status of the autocorrelation function
3. opening and closing of the cardiac fetal valves generate sounds that are registered by the doppler probe. ACF algorithm identifies the associated peaks and generates the reconstructed FHR properly sampled at 2 Hz
4. values of internal or external tocodynamometry
5. fetal motor activity

The data are sampled with a minimum interval of 250ms, but we sampled with an interval of 500ms. After the sampling, the value of the FHR and tocometry are averaged over periods of 2.5s to avoid redundancy. The main limitation of this system is that there are artifacts, as the sensor in old CTGs could record the mother's heart rate instead of the fetal one. The new generation CTGs samples the mother's heart rate together with the FHR to reduce these artifacts. Although there are two phenomenons called:

1. *double-counting*: the FHR is sampled two times and summed, hence simulating a fetal tachycardia

2. *half-counting*: only half of the signals are sampled because the FHR is high and the signals are so close to each other that the transductor is not able to distinguish them.

The transductor is programmed to identify the interval between consecutive R waves. The transductor to discern the signals from the noise coming from the mother's heart rate or the mother's abdomen selects the highest signal closest to the previous R wave, although this can lead to having hidden arrhythmias that are not recognized by the probe due to shape and height. Another scenario might arise when the mother's QRS coincides with a P wave or part of the fetal QRS; the system recognizes it as a correct signal to record, but it contains vertical *spikes*.

To process these signals and remove theirs artifacts, we have used a modified version of the Mantel algorithm, which will be better explained in section 2.2.1

1.2.1 Fetal Heart Rate

The FHR signal is sampled with a frequency of 2Hz and is initially expressed in beats per minute (bpm). We are going to refer to this signal as $F_{120}(i)$ where 120 is the number of points in a minute, while i is the time in which the signal has been sampled. In literature, we might find the FHR expressed in milliseconds (R-R interval); thus, in this case, we will refer to it as $T_{120}(i)$. To calculate the baseline, the program subsamples the signal $F_{120}(i)$ by averaging every 5 points from the starting series, thus getting $F_{24}(i)$. This nomenclature will become useful later on in chapter 3, where it will be used to calculate other essential parameters of the signal.

A typical pattern should have a baseline between 110-160 (bpm); if it is below 110 bpm, then there is bradycardia, while if it is above 160 tachycardia. The FHR is regulated by two Autonomous Nervous System (ANS) components: *parasympathetic* and *sympathetic*. The parasympathetic influences the heart rate by reducing its frequency through the vagus nerve, while the sympathetic increase the baseline. The former develops later than the latter; thus, the FHR baseline decreases as the pregnancy progresses. [4] The most recent guidelines are the FIGO 2015, which are better described in the table 1.1 as with some other guidelines.

When the FHR is between 161-180 bpm for at least 10 minutes of sampling, then we refer to it as tachycardia, and if it is above 180, it is considered highly dangerous. The FHR is highly correlated with either mother's condition or the fetus itself. In fact, mother's hyperpyrexia can shift the dissociation curve of fetal hemoglobin to the right, thus leading to less oxygen for the fetus resulting in *hypoxia*. [5] Also, movements from the fetus or maternal tachycardia influence the FHR.

Instead, when the baseline is between 90-109 bpm for at least 10 minutes of sampling, we refer to it as bradycardia; it is high-risk if it is below 90 bpm. Even this condition can be correlated with the mother's condition or the fetus. For the former, we have autoimmune diseases or prolonged hypoglycemia, while for the latter hypoxia or placental abruption. Lastly, there are forms of idiopathic bradycardia that vanish throughout the pregnancy or at delivery.

The FHR variability is defined as the difference from the highest to lowest frequency in one minute of sampling. It can be described as the amplitude/range of the frequency with values between 6-10 bpm, with the highest at 25, or as the oscillation frequency in a minute with typical values from 2-6 cycles per minute.

The autonomous nervous system, as stated before, regulates the cardiac frequency both with the sympathetic and parasympathetic components. Thus as the pregnancy progresses over time also the variability increases, while instead, in the presence of pathological situations like severe hypoxia, the FHR's variability decreases while at the same time creating dangerous consecutive deceleration. The variability can be measured on three different time scale: short-term (*Short Term Variability*, STV), medium-term (*Delta*), long-term (*Long Term Irregularity*, LTI), which will be defined more in-depth in chapter 3.

Variability hence is a crucial factor in the discrimination of the health status of the fetus. Numerous studies underlined how this discrimination is problematic when considering time intervals greater than 1 minute of sampling. Thus the beat-to-beat variability, which is obtained only through the CTG, is the most accurate parameter to understand the fetus's well-being.[6]

The variability is physiologically reduced during sleep or inactivity of the fetus thanks to the parasympathetic system, but it is healthy only in a time frame of 15-30 minutes, while above 60 is pathological. Therefore if there is an alternation of regions of inactivity with those of typical reactivity, the tracing is not considered pathologic. There are cases in which the variability reduces after a deceleration due to its come back to the baseline. This happens because the deceleration caused a state of transitory hypoxia, which activates

Guidelines		
RCOG 2001	***ACOG 2009***	***FIGO 2015***
Reassuring	**Category I**	**Normal**
Baseline: 110-160	Baseline: 110-160	Baseline:110-160
Variability: ≥ 5 bpm	Variability: moderate	Variability: 5-25 bpm
Decelerations: absent	Late Decelerations or variables: absent Early Decelerations: present or absent	Decelerations: absence of repeated decelerations
Accelerations: present	Accelerations: present or absent	
Non-Reassuring	**Category II**	**Suspicious**
Baseline: 100-109, 161-180	Baseline: Bradychardia / Tachycardia	Lacks at least one characteristics of normality, but with no pathological features
Variability: <5 for 40'-90'	Variability: minimum, absent without repeated decelerations, increased	
Decelerations: premature, variable, 1 prolonged $< 3'$	Decelerations: - repeated variables with minimum or moderate variability - atypical variables - late repeated with moderate variability - prolonged $> 2' < 10'$	
Accelerations: absent	Accelerations: absent after stimulation	
Abnormal	**Category III**	**Pathological**
Baseline: $< 100 > 180$, sinusoidal for $\geq 10'$		Baseline: < 100
Variability: < 5 bpm for $\geq 90'$	Absent variability associted to: 1. Repeated variable decelerations 2. Repeated late decelerations 3. Bradychardia	Reduced variability $> 50'$ Increased variability $> 30'$ Sinusoidal pattern $> 30'$
Decelerations: atypical variables, late 1 prolonged $> 3'$		Late decelerations or repeated prolonged for $> 30'$ or $> 20'$ if reduced variability, 1 prolonged if lasts $> 5'$
Accelerations: absent	Sinusoidal pattern	

Table 1.1: Guidelines

the anaerobic metabolism in the organism's peripheral district, causing a lowering of the peripheral pH, which causes the reduced variability. If this reduced variability persists even after they come back to the baseline, there is a state of prolonged hypoxia. Another cause is $pre - eclampsia$, which is characterized by high blood pressure and some damage to other organs, that associated with an intrauterine slower growth can lead to reduced variability since there is an alteration in the uteroplacental perfusion, thus lowering the oxygen influx in the fetus.

When there is a sudden reduction or absence of variability, it is possible to suppose that there is an artifact. CTG signals contain multiple artifacts which have to be taken into consideration when analyzing the trace. How we processed these signals and their artifacts will be further described in chapter 2. The variability can spike, thus having a *jumping pattern*, when the oscillation's amplitude is greater than 25 bpm as a result of increasing vagal activity. The absence of variability, acceleration, and deceleration in the FHR is also called *silent trace*. It may be due to multiple conditions like Fetal Metabolic acidosis, sleep, inactivity, or arrhythmia. Lastly, there is the sinusoidal trace in which the FHR's oscillations are between the baseline; thus, the variability can be 5-15 bpm, while the oscillation frequency is usually around 2-5 cycles per minute.

Another important parameter when evaluating the FHR is the presence of *accelerations*. Acceleration is referred to as the increase of the FHR of at least 15 bpm above the baseline for more than 15 seconds until 2 minutes, with a later return to the baseline. If it is between 2-10 minutes, it is called prolonged acceleration; otherwise, if it is > 10, there is a baseline variation.

Like the variability, the advancement of the pregnancy influence, due to the further development of the ANS, the amplitude and the frequency of the accelerations. In fact, in pre-term pregnancies or delayed intra-uterine growth, there is a higher presence of small accelerations, in the former is due to the immaturity of the ANS, while for the former, it is caused by the restriction of the movement to save oxygen. [7] The absence of accelerations for more than 30 minutes is usually due to sleep or inaction. [8]

Accelerations can be:

1. *non−periodic*, due to the fetal movement that consumes oxygen and increases the supply from the placenta, it increases its cardiac frequency.

2. *periodic*, happening during uterine contractions that partially compresses the umbilical cord as well as the umbilical vein, causing fetal hypotension activating the baroreceptor response, thus increasing the FHR.

3. component of the variable decelerations due to the compression of the umbilical cord. These types of accelerations are usually components of variable decelerations, and in fact, the latter is most often preceded by a primary or initial acceleration and followed by a secondary one called *overshoot*.

4. followed by a deceleration and takes the name of *lambda pattern*. This is thought to be due to the interruption of the sympathetic activity, with a consecutive activation of the parasympathetic system, thus causing the deceleration, also called *undershoot*.

5. as a result of the *Clark test*: stimulation of the fetus head during labor

Decelerations, instead, are transitory states in which there are transitory slowdowns and periodic of the FHR from the baseline of at least 15 bpm and for more than 15 seconds.

There are two types of decelerations:

1. *uniforms*: have the same shape and are the inverted specular image of the uterine contraction. It is divided in also two subcategories:

   a. *premature* decelerations: lasts at least 20-90 seconds, but the decrement is not greater than 50 bpm. The variability is almost preserved, and the baseline is in the normal range. It is caused by a compression of the fetus's head during labor, producing an increase in the intracranial pressure reducing cerebral blood flow. When this stimulus is ceased, all the values retrace back to normal, hence they are usually linked to a good functioning of the Central Nervous System (CNS). [9]

   b. *late* decelerations: delayed by at least 20 seconds in respect to uterine contractions. The decrement can be as high as 45 bpm, and the variability is reduced too due to a lower oxygen inflow, thus characterizing a pathological status. In fact, during excessive uterine contractions, the fetus reaches a status of hypoxia because there is a reduction of blood flow in the intervillous space. [10] [11]

2. *variables*: different morphologically from the uterine contraction. It is preceded by an initial acceleration and followed by a final acceleration. If it has a normal variability and a normal baseline, it is a sign of good health because it can compensate a hypoxic stimulus. [12] [13]

1.2.2 Tocography

CTG allows getting data of the uterine tone thanks to an external transductor. This technique has its limits, but it suffices for a qualitative valuation since it can pick up:

1. *amplitude*: it depends on the intrauterine pressure. It is around 10-15 mmHg, and usually, it does not go above 30 mmHg
2. *duration*: lasts around 15-20 seconds
3. *frequency*: it varies depending on the gestational weeks. There are the so-called $Braxton - Hicks$ contractions that are irregular, not close to each other, and can stop at any moment. While the labor's contractions happen at a regular time interval.
4. *uterine tone*: it is the uterine pressure between the contractions, the value is between 5-10 mmHg. If the uterus does not go back to its basal tone in between the contractions, this could lower the placental perfusion leading to hypoxia.
5. *rhythmicity*: might comes in the form of couples, triplets, or quartets.

6. *configuration*: usually bell-shaped, but also sheer or pointed aspect during labor. [14]

1.3 State of the Art

This work is relatively new, hence it was challenging to find researches that aim to predict pregnancies' outcome.

Zhao et al. [15] uses FHR signals transformed into images thanks to Continuous Wavelet Transform. Their models consist of an 8-layer Convolutional Neural Networks (CNN) with a single Convolutional Layer. Their dataset is comprised of 2682 and 630 for normal and pathological fetal classes. Even though their model reaches a 98.34% of accuracy with an AUC of 97.82%, the imbalances on their dataset might indicate overfitting. The work of Petrozziello et al. [16] doesn't use images but raw signals from Electronic Fetal Monitoring (EFM) to predict fetal distress. Their approach is to use the FHR and contractions signals to both Long Short-Term Memory (LSTM) and CNN. Results achieved are 61% and 68% respectively. The dataset used was wider as it comprised of 35429 signals, but contained 33959 healthy newborns, while only 1470 compromised. Fergus et al. [17] utilize Machine Learning models to classify caesarean section and normal vaginal deliveries based on cardiotocographic traces. 552 FHR signal recordings, of which 506 controls and 46 pathological, were used as dataset, from which have been extracted features like baseline, accelerations, decelerations, Short-Term Variability (STV) and many others. The models proposed in this paper are: Multi-Layer feedforward neural network, Fishers Linear Discriminant Analysis (FLDA) and Random Forest (RF). The best performing model is the MLP that on the unbalanced dataset reaches an 87% AUC. Lastly, Iraji et al. [18] explores soft computing techniques to predict fetal state using cardiotocogram recordings. Neuro-fuzzy inference system (MLA-ANFIS), Neural Networks and deep stacked sparse auto-encoders (DSSAEs) are implemented. The dataset is composed of 2126 samples already processed that were divided in three classes: 1655 normal, 295 suspect, and 176 pathologic. On the full dataset, the best performing approach is deep learning with an accuracy of 96.7%. The second best is ANFIS that reaches an accuracy of 95.3%.

Chapter 2

Database & Preprocessing

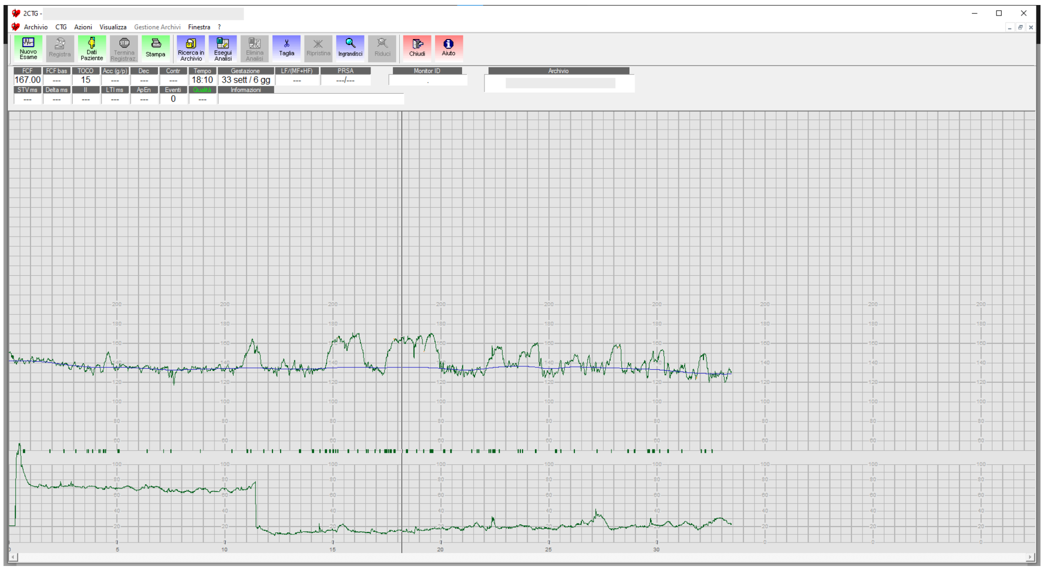
2.1 Dataset

The dataset is taken from the Hospital Federico II of Naples. The cohort is comprised of 9476 pregnant women with 24095 cardiotocographic entries. All the women are from the area of Naples or the neighboring hospitals, and this will play a key role in section 2.1.3 of data labeling.

Pregnancies in this work were binary categorized based on the CTG signal: 0 as healthy and 1 as pathological. The dataset presents a column NOTA, filled by the doctors of the hospital, which contains information about the status of women or pregnancies. This will be further explained in section 2.1.2.

2.1.1 2CTG Software

The software used to process the data is the 2CTG, which is a software developed in the 1993 [2]. It is composed of a series of cardiotocographs that register the CTG trace after it is sent to the computer, where it is stored and displayed on a screen. The data is then sent to another computer containing the software 2CTG, which instantaneously makes an analysis based on linear and non-linear parameters of the FHR. The FHR's tracing is displayed on the top part of the windows with the colors green and yellow influenced by the activity of the autocorrelation function, while when there is a loss of signal is in red. The fetal movements are displayed in the middle part as black dashes right under the FHR's tracing. In the bottom part, there are uterine contractions. The software analyzes the baseline and all the parameters every minute, hence the whole tracing is re-analyzed each time. An example of the 2CTG's user interface is in the following Figure 2.1, some data have been blurred as they contained private information.



FCF	FCF bas	TOCO	Acc (g/p)	Dec	Contr	Tempo	Gestazione	LF/(MF+HF)	PRSA	Monitor ID	Archivio
167.00	---	15	---	---	---	18:10	33 sett / 6 gg	---	---/---	.	

STV ms	Delta ms	II	LTI ms	ApEn	Eventi	Qualità	Informazioni
---	---	---	---	---	0	---	

Figure 2.1: 2CTG User interface with a CTG tracing

HP fetal monitor uses an autocorrelation technique to compare the demodulated Doppler signal of the fetus heartbeat with the next one. Each Doppler signal is sampled at 200Hz. The autocorrelation function is calculated in 1,2 seconds, which corresponds to a minimal FHR of 50 bpm. Hence the software that identifies the peak determines the interval R-R from the autocorrelation function. Interpolation for these peaks is applied and results in a speedup of 2ms.

As of now, the computer can read 10 consecutive values every 2.5 seconds, and the resulting FHR is the mean of these 10 values (corresponding to a sampling frequency of 0.4Hz). The software can sample the FHR at 2Hz (every 0.5s), which is a reasonable value to reach a sufficient bandwidth (Nyquist frequency 1Hz) and a reasonable accuracy.

As stated earlier, there are linear and non-linear parameters for the FHR which will be explained in-depth in chapter 3. [19]

These parameters are:

1. *linear:*

    a. *time domain:*

        i. Baseline
        ii. Short-Term Variability (STV)
        iii. Delta
        iv. Long-Term Irregularity (LTI)
        v. Interval Index (II)

b. *frequency domain*:

i. Low Frequency (LF)

ii. Movement Frequency (MF)

iii. High Frequency (HF)

2. *non-linear:*

a. Approximate Entropy (ApEn)

2.1.2 Data extraction

The dataset was stored in a Microsoft Access Database file (*.mdb*). It contains 6 tables but the important one for this project were only 3: *ANAGRAFE_PCC*, *ESAME_CTG, PARAMETRI_CTG.*

Dataset	***Columns***	***Records***
ANAGRAFE_PCC	14	9879
ESAME_CTG	19	29353
PARAMETRI_CTG	73	24095

Table 2.1: Records example in ANAGRAFE_PCC table

The tables contained the following features, and only the most important one will be listed:

1. *ANAGRAFE_PCC:*

i. ***PATNUM***, Patient number used as a key to referencing each patient

ii. ***NOME***, Patient's name

   iii. ***COGNOME***, Patient's surname
   iv. ***DATA_NASCITA***, Date of birth
   v. ***NOTA***, Notes related to the health status of patient or if is coming from another hospital
   vi. ***CITTA***, City from which the patient comes from

2. *ESAME_CTG:*
   i. ***PATNUM***
   ii. ***ID_PRESTAZIONE***, The ID linked to a CTG tracing
   iii. ***DATA***, date and time in which the CTG was recorded
   iv. ***SETT_GESTAZIONE***, Gestation week

3. *PARAMETRI_CTG:*
   i. ***ID_PRESTAZIONE***
   ii. ***STV_MED***, Short-Term Variability
   iii. ***II_MED***, Interval Index
   iv. ***DELTA_TOT***
   v. ***LTI_MED***, Long-Term Irregularity
   vi. ***APEN_MED***, Approximate Entropy
   vii. ***LF_MED***, Low Frequency
   viii. ***MF_MED***, Movement Frequency
   ix. ***HF_MED***, High Frequency

x. ***RC_MED***,

xi. ***NUM_ACCEL_GRANDI***, Number of big accelerations

xii. ***NUM_ACCEL_PICCOLE***, Number of small accelerations

PARAMETRI_CTG contains for all the features listed above, their respective Minimum, Maximum, and quantile values, except for ID_PRESTAZIONE. These features were excluded for the sake of brevity. The visual structure of these tables can be find in the tables 2.1, 2.2 and 2.3 respectively.

In order to link the data contained in the dataset tables, we used the women's personal information. To calculate the age of patients, as it was not provided in the dataset, we had to extract the date on which the CTG recordings have taken place. Lastly, we extracted all the information regarding the CTG trace.

We used Python with the module pandas to perform all the data extraction and processing. A more thorough explanation of the tools and libraries used is in section 2.3.

In order to get the information of each woman, we performed a join between ESAME_CTG, PARAMETRI_CTG, and ANAGRAFE_PCC on ID_PRESTAZIONE between the first two tables and on PATNUM between the first and the last table.

The resulting MSQL query is the following:

```
SELECT ANAGRAFE_PCC.NOME, ANAGRAFE_PCC.COGNOME,
ANAGRAFE_PCC.DATA_NASCITA, ANAGRAFE_PCC.NOTA,
ANAGRAFE_PCC.CITTA, ESAME_CTG.DATA, ESAME_CTG.QUALITA,
ESAME_CTG.SETT_GESTAZIONE, PARAMETRI_CTG.*

FROM ESAME_CTG, PARAMETRI_CTG, ANAGRAFE_PCC

WHERE ESAME_CTG.ID_PRESTAZIONE=PARAMETRI_CTG.ID_PRESTAZIONE and
ANAGRAFE_PCC.PATNUM=ESAME_CTG.PATNUM
```

The query returned a total of 24095 records.

All the data shown in the following tables are not representing any real person.

PATNUM	***NOME***	***COGNOME***	***DATA_NASCITA***	***NOTA***	***CITTA***
5	Chiara	Rossi	14/08/1984	04 iugr	
28	Giulia	Mareschini	19/04/1990	pz prov unità remota	Afragola

Table 2.2: Records example in ANAGRAFE_PCC table

PATNUM	***ID_PRESTAZIONE***	***DATA***	***SETT_GESTAZIONE***
5	14	6/2/2018	32
5	29	18/2/2018	34

Table 2.3: Records example in ESAME_CTG table

ID_PRESTAZIONE	***STV_MED***	***II_MED***	***DELTA_TOT***	***LTI_MED***	***APEN_MED***	***LF_MED***	***MF_MED***	***HF_MED***	***RC_MED***	***NUM_ACCEL_GRANDI***	***NUM_ACCEL_PICCOLE***
14	5,53	0,85	36,18	24,44	1,24	84,17	13,63	2,18	4,72	6	0
22	3,71	0,52	41,31	26,13	1,59	81,31	12,59	1,84	3,97	20	3

Table 2.4: Records example in PARAMETRI_CTG table

2.1.3 Data Labeling

First of all, the main focus was on the column NOTA that contained a code, given by the hospital Federico II, for each type of category in which each woman/pregnancy would fall into. The categories were as follow:

01: Physiological

02: Twinning physiological

03: Fetal Pathology:

  (a) Intra Uterine Growth Restriction (IUGR)

  (b) Flowmeters alterations

04: Maternal Pathology

  (a) Diabetes type I

  (b) Diabetes type II

  (c) Gestational Diabetes

  (d) Essential hypertension

  (e) Drug addiction

  (f) Cardiopathy

  (g) Thyroid

05: Fetal malformation

06: Fetal pathology + maternal pathology

07: Fetal pathology + fetal malformation

08: Maternal pathology + fetal malformation

09: Twinning + fetal pathology

010: Twinning + maternal pathology

011: Twinning + fetal malformation

012: Gestational week < 30

In this study, we divided the dataset into healthy, 01 and 02, and pathological, from 03 to 12. We were not interested in discerning all the different pathologies but rather in understanding if the pregnancy was healthy or not.

The resulting dataset needed to be skimmed through because in the column NOTA, there was another category which was *"pz da unità remota"* or *patient coming from remote unit* and it did not contain the category's code. To understand how to categorize the woman/pregnancy, Federico II Hospital provided a list containing these neighboring hospitals with the correlated category, thus helping us discerning if the patient was healthy or pathological.

We extracted a blob file from the column PARAMETRI in the table PARAMETRI_CTG which had essential parameters like the FHR, TOCO, and decelerations of a specific CTG tracing. The blob was then passed through the software 2CTG that extracted these parameters. We needed to merge these records with the dataset from ESAME_CTG to retrieve only the women who had an available CTG tracing. To accomplish this, we had to use a naive solution because the blob file did not contain any information about the original patient, and for how the 2CTG works, there was not an easy way to link these types of data. After inspecting how the files were processed by the 2CTG, we understood that they had the same order as the one from the database; more specifically, they were ordered by the ID_PRESTAZIONE column. Hence we intersected the list of ID_PRESTAZIONE from ESAME_CTG with the list of ID_PRESTAZIONE from PARAMETRI_CTG. The resulting dataset was cleaned of patients that: had an invalid CTG tracing (< 2400 points), did not contain a CTG tracing, and contained null values in the columns NOME, COGNOME, NOTA, SETT_GESTAZIONE, DATA_NASCITA, resulting in a total of 17483 valid records.

We derived two different datasets from this dataset: one containing only the overall mean values of the CTG tracing, while the other contained all the values calculated by the original CTG system.

2.2 Pre-Processing

2.2.1 Signals correction and baseline extraction

Most CTG signals contain artifacts, as stated in chapter 1, due to an overlap of the mother heart rate and the FHR, sudden movement by the mother or fetus, or simply an impossibility of the transductor to sample the signal. Although artifacts are expected in a CTG signal, they are not valid for machine learning training.

To solve this issue we used a modified version of the Mantel's algorithm to extract the baseline[20] and it is contained in the 2CTG software. Before this algorithm the baseline estimation was done in intervals and thus losing some signal's oscillations. An example of a before and after the algorithm can be seen in Figure 2.2 and 2.3.

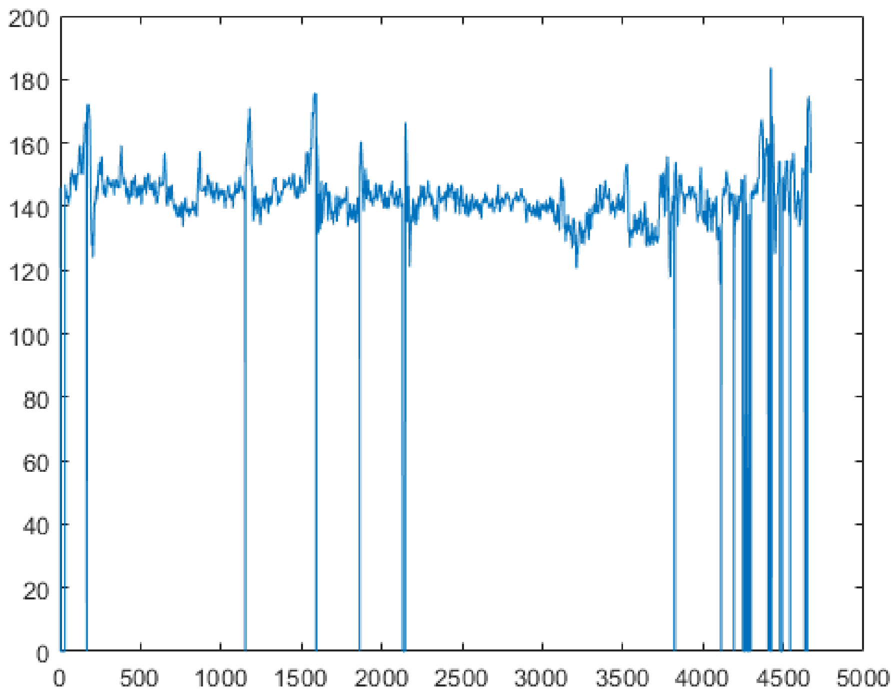


Figure 2.2: Example of a signal containing artifacts

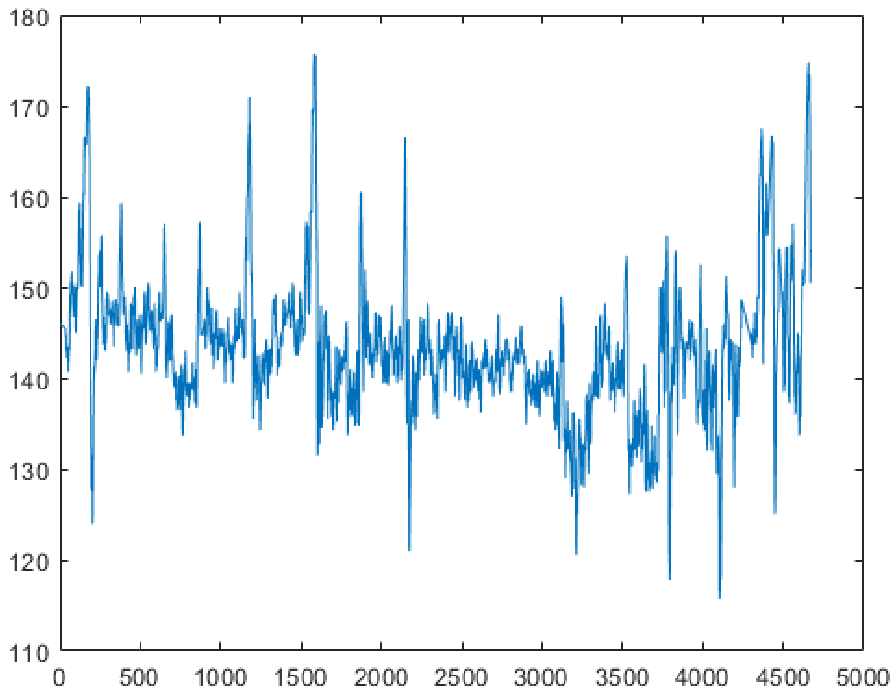


Figure 2.3: Example of a signal after the correction

2.2.2 Final dataset

After all the signals were corrected, we had to create one last part of the dataset, which were the images used for the CNN part of our work. From our initial 17483 valid records, we first split each FHR signal into a sub-sequence of 20 minutes that contains 2400 sampled data (the minimum to be considered valid) with an overlapping algorithm. These sub-sequences were then converted into images of different sizes (32x32x3, 64x64x3, 256x256x3) using multiple techniques:

1. Wavelets
2. Gramian Angular Summation Field (GASF)

3. Gramian Angular Difference Field (GADF)
4. Markov Transition Field (MTF)
5. Recurrence Plot (RP)
6. Self-Similarity Matrix (S)
7. Power Spectrogram (PS)
8. Persistence Spectrum (PSP)

The mathematical background and theory behind these techniques will be further explained in section 4.3.1. The final images dataset was then comprised of 48398 images, of which 20588 were healthy, while the remaining 27810 were pathological.

2.3 Tools & libraries

Our work used popular libraries for Machine Learning/Deep Learning:

1. *Python 3.7.6*: was the primary language used to perform most of the data extraction, pre-processing, training of the machine learning models. The required libraries were:

   - *Numpy 1.19.2* is the most commonly used library to compute operations between multi-dimensional arrays.
   - *Pandas 1.1.5* is the most famous library to process and manipulate tabular data.

- *Scikit-learn 0.24.2* is a simple and efficient set of tools for predictive data analysis. It has a variety of Machine Learning models.
- *Scipy 1.4.1* is used to perform statistical analysis and computations. Used only to load Matlab (.mat) files.
- *Keras 2.4.3* is one of the most common Deep Learning APIs for python
- *Tensorflow 2.4.1* is a ML library that contains Deep Learning modules used for the most complex models pertaining to Neural networks.
- *Matplotlib 3.4.1* was used for data visualization
- *Seaborn 0.11.1* is another data visualization tool in order to have more complex and decorated plots
- *Pyodbc 4.0.30* was used to interact with Microsoft Database file (.mdb)
- *Xgboost 0.71* a library to use this model

2. *Matlab R2021a*: was the main programming language used to operate on signals since it is easy to perform operations on multi-dimensional arrays and to convert signals into images. Also, it was needed because the implementation of the Mantel algorithm was in Matlab.
3. *Jupyter Notebooks 6.4.0* for interactive execution and results' display

To train and test our models, we used Kaggle, a website that allows users to upload datasets, implement machine learning models, and test them using their hardware.

Chapter 3

Machine Learning

In this chapter, we start with introducing CTG's features and how they are calculated, proceeding with the machine learning algorithms adopted and the results achieved.

3.1 CTG's Features

CTG's systems, like already stated in chapter 1, calculate various parameters that are crucial for doctors in understanding fetus's status. We will re-use the annotation, firstly described in chapter 1, of the signal:

1. $F_{120}(i)$: normal signal with 120 points in a minute
2. $T_{120}(i)$: signal when expressed in milliseconds (R-R interval)
3. $F_{24}(i)$: subsampled signal by 5 points
4. $T_{24}(i)$: subsampled signal by 5 points, but in milliseconds

3.1.1 Time Domain Parameters

Short-Term Variability

It is employed to quantify the FHR's variability over a short temporal scale. The definition explained here comes from Dalton et al. [21] and Arduini et al. [2].

Considering an interbeat sequence of 1 minute, $T_{24}(i)$ in ms, where $i = \mathit{1,\ldots,24}$, the STV is defined as:

$$\mathbf{STV} = \mathbf{mean}\left[|T_{24}(i+1) - T_{24}(i)|\right]_i = \frac{\sum_{i=1}^{23} |T_{24}(i+1) - T_{24}(i)|}{23}, i = 1, ..., 23$$

where $T_{24}(i)$ is sampled every 2.5s.

The STV is calculated each minute on signal segments of 3 minutes, excluding the FHR's periodic variation like accelerations and decelerations.

It is the most used parameter for a proper clinical evaluation of fetuses with intrauterine growth retardation, as it correlates with the degree of fetal hypoxia/acidosis and with the neonatal outcome at birth [22]. In fact, abnormal values are linked to drastic changes in fetal conditions, increasing the risk of delayed motor and neurological development while also damaging specific brain areas with long-term cognitive effects. This parameter is crucial for fetal homeostasis, especially for medium/long-term variability, because it checks the ANS's integrity and its connections with the CNS. When the STV is high, the ANS is healthy and functioning correctly; low STV is associated with an oxygen detriment, thus fetal distress. [23] [24]

Long-Term Irregularity

The LTI proposed by De Haan et al. [25] measures the global trend of the frequency of long-term fluctuations. It can be either expressed in terms of range or frequency. The former is calculated starting from the FHR's baseline over one minute of recording while noting its highest and lowest point. The usual variability range is between 5 and 20 bpm. The latter, instead, is measured by counting the number of FHR's fluctuations in 1 minute of recording. A typical range is between 2-6 fluctuations; lower than 2 is considered abnormal.

The LTI is measured over interbeat sequences of 3 minutes each in milliseconds, but it does not account for big accelerations and decelerations, as suggested by Arduini et al. [2], to avoid deviation caused by spurious measurements of variability. The whole sample has to contain a continuous segment of at least 30 seconds.

With a given signal $T_{24}(i)$, LTI is the interquantile interval

$$\left[\frac{1}{4}; \frac{3}{4}\right]$$

of the distribution $m_{24}(j)$ with $j \in [a; b-1]$:

$$m_{24}(j) = \sqrt{T_{24}^2(j) + T_{24}^2(j+1)}$$

Delta

Given a 1 minute signal in milliseconds $T_{24}(i)$ with $j \in [1; 24]$, Delta is the difference between the max and the min values of the FHR in a 1-minute recording:

$$Delta = \max T_{24}(i) - \min T_{24}(i)$$

In [2] the big accelerations and decelerations are excluded.

Interval Index

It is one of the most widely used indices. Firstly proposed by Yeh et al.[26] as a long-term variability parameter. The system 2CTG, though, uses the II proposed by Arduini et al. [2].

$$II = \frac{std\left[T_{24}(i+1) - T_{24}(i)\right]}{STV}$$

where STV stands for Short-Term Variability.

The Interval Index is a variation coefficient of the differences between all the FHR's values in a 1-minute recording, averaged over periods of 2.5 seconds.

3.1.2 Frequency Domain Parameters

The ANS controls the FHR's variability, thus it is essential to quantify its development during pregnancy. The Power Spectral Density (PSD) is used to gather information related to the activity of the ANS on the FHR's. Three frequency bands are considered: Low Frequency (LF), Movement Frequency (MF), and High Frequency (HF). [3]

Low Frequency

The LF is influenced by the baroceptor feedback activity that is linked to the oscillations of blood pressure and mediated by the sympathetic activity of the ANS, while according to some authors, it includes both sympathetic and parasympathetic activity. The range for the LF is 0,03-0,15 Hz.

Movement Frequency

The MF considers the fetal movements, specifically of the trunk, correlating with the mother's breathing frequency. It works on a small time frame of about 3-5 minutes, approximately 300 data points. The range for the MF is 0,15-0,5 Hz.

High Frequency

The HF represents the effect that breathing has on the sinus node and is mainly associated with the parasympathetic activity of the ANS. The range is 0,5-1 Hz.

3.1.3 Non-linear Parameters

The use of a non-linear approach allows the usage of methods that study the geometric and dynamic properties of the temporal series, although only estimations can be made on these parameters, which results in being important.

Approximate Entropy

The Approximate Entropy (ApEn) quantifies the complexity (irregularity) of the FHR's variability on windows of 3 minutes each. Low ApEn values indicate a lower complexity of the signal and vice versa. Li et al. indicate that lower values of ApEn are associated with fetal distress, hypoxia, and respiratory and metabolic acidosis in women close to labor. [27]. The definition is taken from Pincus [28].

$$ApEn(m,r) = \frac{\sum_{i=1}^{N-m+1} \log C_i(m,r)}{N-m+1} - \frac{\sum_{i=1}^{N-m+1} \log C_i(m+1,r)}{N-m}$$

It works specifically for short and noisy temporal series, less than 100 samples. N is the fixed length of the experimental temporal series, m is the length of the segment in respect to the series, r is the signal's percentage std (that works as a filter), and C_i is the frequency of similar patterns given a pattern in the window m.

3.2 Feature Selection

We used multiple feature selection methods to eliminate redundant or irrelevant variables. These techniques are 6: 2 filter methods, 2 embedded methods, 1 wrapper method, and the last one is dimensionality reduction. We also used a Variance threshold to discard less meaningful features.

1. **Univariate Feature Selection (UFS)** [29], uses univariate tests to extract the most relevant features
2. **Recursive Feature Elimination (RFE)** [30], recursively eliminates features with the least importance
3. **Random Forest Feature Selection (RFFS)** [31], selects the most important features according to random forest scoring
4. **Decision Tree (DT)** [32], can be used also for feature selection. In fact it selects the most important features, by keeping them close to the root of the tree
5. **Lasso Regularization** [33], shrinks less important features to zero
6. **Principal Component Analysis** (PCA) [34], reduces dimensionality of the dataset and minimizes information loss

3.2.1 Principal Component Analysis

We talk more in-depth about PCA than the other techniques, as it returns not a list of selected features but a list of Principal Components that explains the most variance inside the dataset.

The following table has been created to make more readable the subsequent graphs:

Feature	**ID**
SETT_GESTAZIONE	1
STV_MED	2
II_MED	3
DELTA_TOT	4
LTI_MED	5
FHRB_MED	6
APEN_MED	7
LF_MED	8
MF_MED	9
HF_MED	10
RC_MED	11
NUM_ACCEL_GRANDI	12
NUM_ACCEL_PICCOLE	13
AGE	14
DEC	15

Table 3.1: Features' ID

We first plotted the dataset and the Principal Components found by the algorithm. The arrow have been scaled up by a factor of 1.15 for readability purposes.

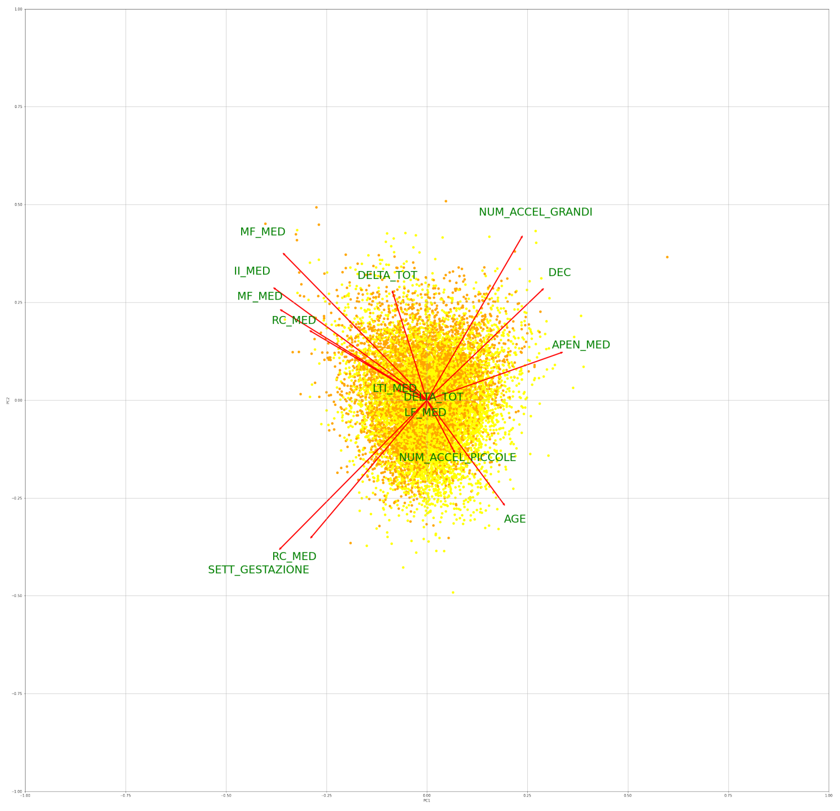


Figure 3.1: Plotted dataset and its principal components

From figure 3.1 we can already understand which PC explains the most variance, but we plotted the explained variance for each feature.

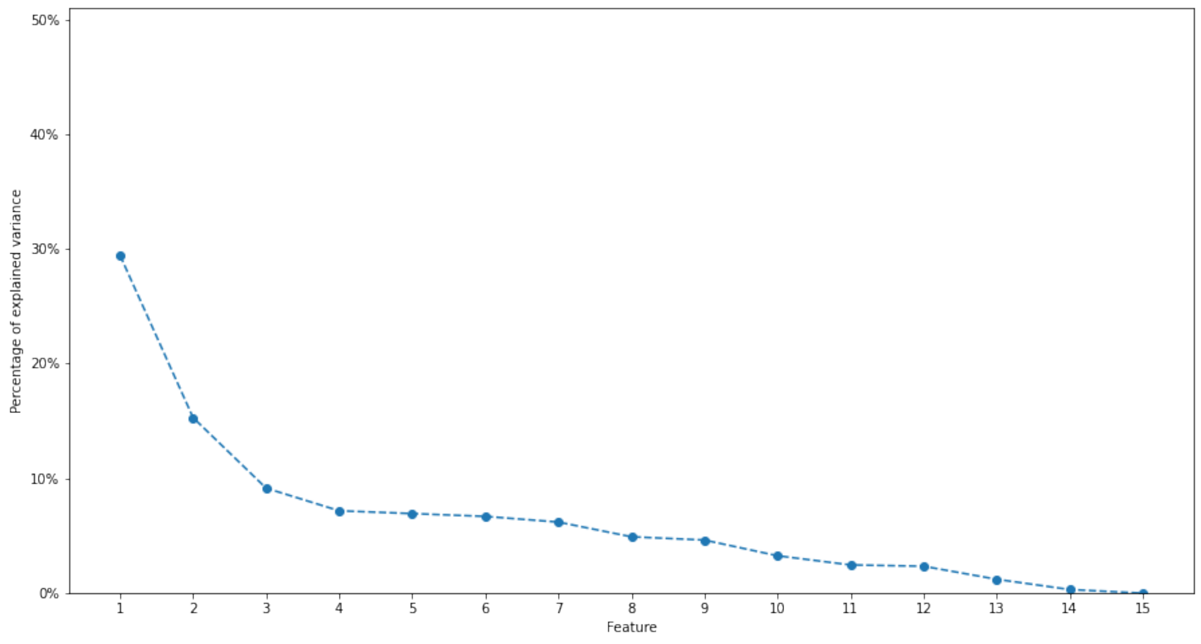


Figure 3.2: Explained variance by each feature

We set the threshold for the PC explained variance 90%. To understand which were the most crucial PC, we used the chart described in figure 3.3. We can see clearly how 9 features explain 90% of the variance. Based on this result, we trained our model, and the results can be seen in the next section in the table 3.3

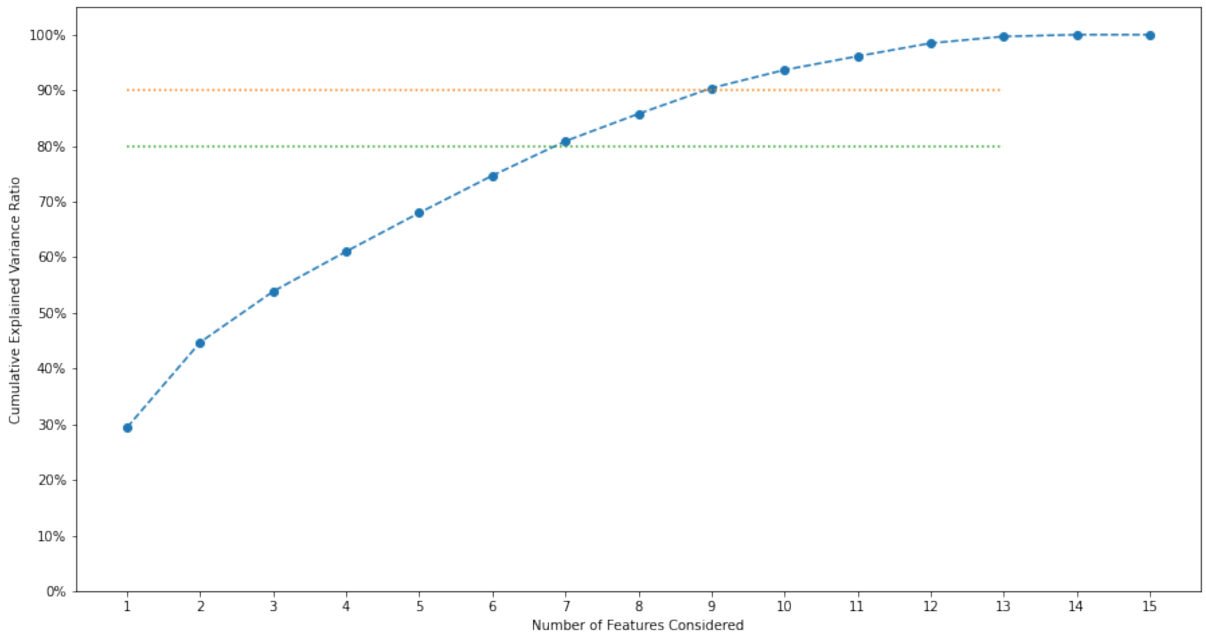


Figure 3.3: Cumulative explained variance

3.3 Feature Scaling

Our dataset contained features with different scales. To avoid any feature obscuring another, we applied normalization because we did not know the distribution of our data. We used the MinMax method to scale all the features between the range (0,1).

3.4 Hyperparameters Tuning

We used Randomized Search with 5-fold cross-validation [35] to find the best hyperparameters for each of our models. In section 3.5 will be described more in-depth models' specifications.

3.5 Algorithms

This section introduces the algorithms used in our work that had to binary classify our dataset (healthy/pathological).

1. *Logistic Regression (LR)* [36] is used to model the probability of a certain class using the logistic function.

2. *Random Forest (RF)* [37] utilizes ensemble learning, which is a technique that, by combining many decision trees (classifiers), chooses the class selected by most trees.

3. *K-Nearest Neighbors (k-NN)* [38] is a supervised non-parametric classification method. It works based on voting: a new sample is classified by a plurality vote of its neighbors (the closest training examples based

on a distance metric). The class chosen is the most common among its k nearest neighbors. It is a lazy learner as it will not remember the model and make all the calculations at run time; because of this it is considered memory-intensive.

4. *Adaboost* [39] is a ML meta-algorithm based on the concept of boosting: weak learners are trained, and their output is combined into a weighted sum which represents the output of the boosted classifier. It reduces drastically overfitting [39].

5. *Multi Layer Perceptron (MLP)* is a type of Neural Network that can recognize underlying correlations in a dataset. We used 4 hidden layers with L2 regularizers, as loss function, we used binary cross-entropy, the optimizer chosen was Adam, and the sigmoid function as the activation function.

6. *Support Vector Machine (SVM)* is a supervised learning model with associated learning algorithms that tries to maximize the width of the gap between the two categories by mapping points in space. When there is a new data point, the SVM checks where it belongs into that space and classifies it depending on which side of the gap it fell.

7. *Bagging* is a machine learning ensemble meta-algorithm that helps in improving the stability and accuracy of machine learning algorithms. Like boosting, it helps in reducing overfitting, but Bagging also reduces the variance. It generates multiple new training sets, each of a specific size n, by sampling from the original dataset uniformly and

with replacement.

8. *GradientBoosting* It works similarly to the other boosting methods. It combines weak "learners" into a single strong learner in an iterative way.

9. *Extreme Gradient Boosting (XGB)* [40] is one implementation of Gradient Boosting, but it is more regularized than its ancestor Gradient Boosting. XGB uses advanced regularization (L1 & L2), which improves model generalization capabilities. It is notably faster and has better performance than Gradient Boosting, while it can also be parallelized.

All the hyperparameters were extracted using Randomized Search with 5-fold cross-validation [35]. For the reference model "Logistic Regression" we used L2 regularization with C equal to 10, using *liblinear* and *lbfgs* as solvers. For the *Support Vector Machine* we uses C equal to 13 with *rbf* as kernel. For the*Multi-Layer Perceptron* we used 4 hidden layers, respectively: 500, 250, 150, 50. For the *k-NN* the neighbours were 5, with metric Manhattan distance. For the *Bagging*, the estimators chosen was the SVM, with 20 estimators. For the *Gradient Boosting* the estimators were 200 with a learning rate of 0.1. For *AdaBoost* the estimators were 100. For the *XGB* we used 100 estimators with learning rate of 0.1. For the *Random Forest* we used 100 estimators with entropy as the criterion.

3.6 Results

We trained multiple models to understand which performed better. Before testing our algorithms, we used a variance threshold that helped us to take out features as *ApEn* and *II* as they had low variance, thus low statistical significance. In Table 3.2 we show the features selected by each technique, but we omit PCA as explained earlier. In Table 3.3 we show the results of our trained models. To have a clearer overview of our results, we added:

1. **Recall**, also known as sensitivity, indicates the percentage of pathological data point in the test set that the model was able to label as such

2. **F1-Score**, is the harmonic mean of recall and precision, so it adds extra information that recall alone does not provide

3. $\mathbf{R^2}$, explains the proportion of the variation in the dependent variable from the independent variable,

4. **Root Mean Squared Error (RMSE)**, is the standard deviation of the prediction errors. These errors are a measure of how far from the regression line data points are.

5. **Area Under the Curve (AUC)**, is an integrated measurement of a measurable effect or phenomenon. Describes how well the classifier can distinguish between the two classes. It is calculated by doing a definite integral between two points.

3.7 Limitations

Although Machine Learning algorithms can learn to discriminate different datasets in complex planes, we understood that the results were not great and that only numerical data were not enough to understand how to classify our dataset. Our idea was to combine the MLP with the power of Deep Learning, using images and temporal series. In the next chapter, we dive further into this topic, explaining how we performed and achieved improvements.

Models	***Features Selected***	***Discarded Features***
Variance Threshold	*SETT_GESTAZIONE, STV_MED, DELTA_TOT, LTI_MED, FHRB_MED, LF_MED, MF_MED, HF_MED, RC_MED, NUM_ACCEL_GRANDI, NUM_ACCEL_PICCOLE, AGE, DEC*	*II_MED, APEN_MED*
Univariate Selection	*SETT_GESTAZIONE, STV_MED, DELTA_TOT, LTI_MED, FHRB_MED, HF_MED, NUM_ACCEL_GRANDI, NUM_ACCEL_PICCOLE, AGE, DEC*	*II_MED, APEN_MED, LF_MED, MF_MED, RC_MED*
Recursive Feature Elimination	*SETT_GESTAZIONE, FHRB_MED, HF_MED, NUM_ACCEL_PICCOLE, AGE*	*II_MED, APEN_MED, STV_MED, DELTA_TOT, LTI_MED, LF_MED, MF_MED, RC_MED, NUM_ACCEL_GRANDI, DEC*
Random Forest Feature Selection	*SETT_GESTAZIONE, STV_MED, DELTA_TOT, LTI_MED, FHRB_MED, LF_MED, MF_MED, HF_MED, RC_MED, NUM_ACCEL_GRANDI, II_MED, APEN_MED, AGE*	*DEC, NUM_ACCEL_PICCOLE*
Decision Trees	*SETT_GESTAZIONE, STV_MED, II_MED, APEN_MED, DELTA_TOT, LTI_MED, RC_MED, FHRB_MED, LF_MED, MF_MED, HF_MED, AGE*	*NUM_ACCEL_GRANDI, NUM_ACCEL_PICCOLE, DEC*
Lasso Regularization	*SETT_GESTAZIONE, DELTA_TOT, FHRB_MED, LF_MED, NUM_ACCEL_GRANDI, AGE*	*II_MED, APEN_MED,STV_MED, LTI_MED, MF_MED, HF_MED, RC_MED, NUM_ACCEL_PICCOLE, DEC*

Table 3.2: Features Selected by each algorithm

Models	***Best Model***	***R-Squared***	***RMSE***	***F1-Score***	***Recall***	***AUC***
Whole Dataset	XGB	0.697	0.561	0.68	0.68	0.685
Variance Threshold	XGB	0.699	0.563	0.68	0.68	0.682
Univariate Selection	RF/XGB	0.7	0.56	0.68	0.68	0.682
RFE	XGB	0.692	0.569	0.67	0.67	0.672
Random Forest	Bagging	0.687	0.566	0.68	0.68	0.676
Decision Trees	SVM	0.695	0.562	0.68	0.69	0.688
Lasso	/	0.164	/	/	/	/
PCA	Bagging	0.689	0.569	0.67	0.68	0.679

Table 3.3: Classifiers trained on datasets with different features

Chapter 4

Deep Learning

In this chapter, we explain our approach using Neural Networks (NN), in the specific both Multi Layer Perceptron (MLP) and Deep Learning (DL), and how we managed to achieve better results. In the specific, we first make a general introduction on Neural Networks, then focusing on the architectures used in our work: MLP, Convolutional NN (CNN), and Long Short-Term Memory (LSTM). Before diving into the CNN, we explain the images extracted from the signals and the theory behind them. We want to compare these techniques related to the Machine Learning approach by comparing their accuracy and how we can improve our architecture.

4.1 Introduction to Neural Networks

As the Machine Learning's results were stuck to a local optimum of 70%, we opted in trying a new pioneering hybrid approach. We used Neural Networks, also known as artificial neural networks (ANNs), which are a subset of machine learning and are the core of deep learning algorithms. Their name and structure are inspired by the human brain and how it learns, mimicking how its neurons communicate with each other.

Neural Networks are versatile, and their applications are multiple. They can be used for classification, as in this work, regression, object detection, speech recognition, and many other domains. Their recent success is due to Deep Learning, a subset of machine learning, which is essentially a neural network with three or more layers. The need to create a deeper structure was suggested by Bengio and Delalleau [41]. They thought it is natural to search for a deeper structure since the human neural system is a deep architecture and that humans tend to represent abstract concepts using multiple lower-level concepts.

The linear perceptron could not be a universal classifier, but when a hidden layer with unbounded width and nonpolynomial activation function is added to a network, creating a deep architecture, it can become a universal classifier. Deep learning discovers complex structures [42] in large data sets by using the backpropagation algorithm that tune the internal network parameters of the previous layers.

In Deep Learning, there are multiple architectures:

1. **Convolutional Neural Networks (CNN)**, use kernels and filters to slide through images, videos, or time series to learn to classify them. They are widely used in image and video recognition, image classification, image segmentation, medical image analysis, and other domains. CNNs were inspired by biological processes as their organization resembles the visual cortex. In-

dividual cortical neurons respond to stimuli only in a specific visual field region known as the receptive field. The receptive fields of different neurons partially overlap, covering the entire visual field.

2. **Recurrent Neural Networks (RNN)**, use their "memory" as they take information from prior inputs to influence the current input and output. Deep neural networks assume that inputs and outputs are independent of each other, while recurrent neural networks' output depends on the prior elements within the sequence. They are used temporal problems, natural language processing, speech recognition, and image captioning.

3. **Long Short-Term Memory (LSTM)**, is an RNN architecture. It has feedback connections. It is used primarily on time series data. LSTMs were solves the problem of the vanishing gradient problem that pertains to traditional RNNs.

4. **Autoencoders**, they pertain to the unsupervised learning field as they learn unlabeled data. [43] The autoencoder learns a representation of the data; it ignores the noise and applies a dimensionality reduction as it needs to have a compressed knowledge of the input.

At the core of each ML technique is the amount of data that will be fed as input to these algorithms. We expect the model to generalize from these known examples, which is a challenging task. To accomplish this, the dataset is split between train/test and applied k-fold cross-validation. The most common scenario is that the model overfits. Usually, this happens because the model has been trained too much on the training dataset and can learn noise and details that negatively impact its ability to discriminate.

4.1.1 Avoid Overfitting

Overfitting is a crucial problem to overcome in Machine Learning. Especially in Neural Networks. In fact Multi Layer Perceptrons are composed of fully connected layers, where each neuron in one layer is connected to all neurons in the subsequent layer. This ”full connectivity” that characterizes these networks makes them prone to overfitting data. To solve this problem, there are regularization techniques that prevent overfitting:

1. **Weight Decay**, which is similar to L_2 regularization, every weight is multiplied by a factor w_d, where $0 < w_d < 1$

2. **Batch Normalization**, [44] it re-scale and re-center layer's input to make the network more stable and faster.

3. **Dropout**, [45] it avoids complex co-adaptations on training data. To accomplish this some neurons are ”dropped out” or turned off completely.

4. **Skip Connections**, allows to skip certain layers in the architecture and the output of one layer is fed as input to another layer (instead of only the next one). As cited in Drozdzal et al. [46] the model that had skip connections performed better and had a faster convergence.

In our work, we used Batch Normalization and Dropout. We penalized our CNN heavily, with Dropout reaching almost 80%, while for the MLP we had 40%. We used weight decay only in one layer of the MLP as it boosted the accuracy by 5%.

4.2 Multilayer Perceptron

As stated earlier, the MLP is a fully connected network. It is helpful as it allows approximate solutions for very complex problems. We have used it as a classifier to discern if a pregnancy was pathological or not by giving it as input the numerical data about the mother's personal details and CTG's signal.

Neurons uses activation function, just like the action potentials of biological neurons. There exists different activation function:

1. *Rectified Linear Unit (ReLU)*: is a non-linear function, also known as ramp function

$$f(x) = \max(0, x)$$

2. *Leaky ReLU*: solves the problem of the dying neurons of the ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases}$$

3. *Sigmoid*: mainly use to predict the probability to pertain to a specific

class, since it exists between $(0, 1)$

$$S(x) = \frac{1}{1 + e^{-x}}$$

4. *Softmax*: is a generalization of the logistic function, and it is used at the end of the neural network to normalize the output to a probability distribution over a set of classes

$$\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

In our network, we used ReLU and the Softmax function, as they achieved the best results. The network is a sequential model that contains 4 hidden layers with an output layer to binary classify.

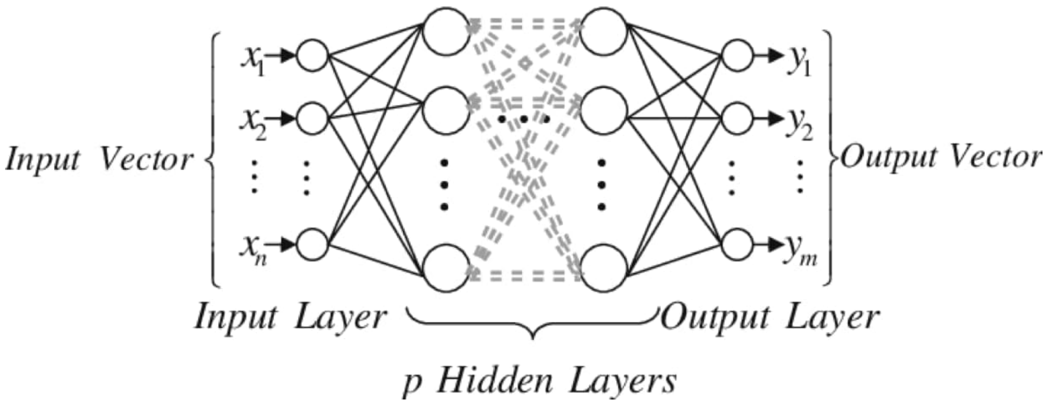


Figure 4.1: Example of a Multi Layer Perceptron

4.2.1 Our Model

The structure of our MLP is as follow:

1. **Input**: 16 features
2. **Hidden Layer 1**: 500 neurons, activation function ReLU
3. **Dropout**: 40%
4. **Hidden Layer 2**: 250 neurons, activation function ReLU
5. **Dropout**: 40%
6. **Hidden Layer 3**: 150 neurons, activation function ReLU with kernel regularizers both L1 and L2 regularization penalty, bias and activity regularizers L2. The value for L1 is $1e^{-5}$, for L2 we have $1e^{-4}$.
7. **Hidden Layer 4**: 50 neurons, activation function ReLU
8. **Dropout**: 40%
9. **Output Layer**: 2 neurons, activation function Softmax

We have used as optimizer Adam with learning rate 10^{-4} and decay $\frac{10^{-4}}{200}$. The loss function used is the binary cross-entropy.

4.2.2 Results Achieved

The dataset used for the MLP contains the mean values of the following features for each recording: DELTA, II, STV, LTI, APEN, LF, MF, HF, RC, FHRB, except for Small accelerations, Big accelerations, Decelerations, Gestational Week and Mother's age. The recording to be considered valid had to contain at least 2400 points; otherwise, it got discarded.

The dataset was split between training and test (80%/20% respectively), and our model was subsequently trained.

The accuracy reached by our model was 75.7%.

4.3 Convolutional Neural Networks

This section explains the images we used and how we trained our classifier on the new dataset composed of images using Convolutional Neural Networks (CNN), and the results achieved.

4.3.1 Images Transformation

In order to use the CNN with images, we had to transform our signals into images that our network could read. Signal to image transformations are becoming more common as deep learning recent successes in computer vision inspire different methods to approach old problems. We can classify a time series transformed into an image by exploiting the techniques of computer vision.

We decided to use multiple techniques to have a richer dataset that tried to generalize the problem. Our dataset was comprised of images of dimensions: 32x32x3, 64x64x3, and 256x256x3. We wanted to test on different dimensions to see how much it would have impacted our results. All the images have been extracted using Matlab, and some were created thanks to built-in functions of Matlab, while others were algorithmically created by us, using as reference the papers cited.

Continuous Wavelets Transform

The Continuous Wavelet Transform (CWT) [47] is used to decompose a signal into wavelets. Wavelets are small oscillations that are highly localized in time. Wavelet decomposition is performed by analyzing or so-called "mother" wavelet, a time-localized oscillatory function. The mother wavelet is continuous in both time and frequency and is the source function that constructs the scaled and translated basis functions. The CWT constructs a time-frequency representation of a signal that has a good time and frequency localization. Figure 4.2 represents one of the wavelets images from our dataset.

The CWT performs well in mapping the changing properties of non-stationary signals and determines if the signal is overall stationary. It is a convolution of the data sequence with a scaled and translated version of the mother wavelet. The convolution can be accomplished directly, as in the first equation, or via the FFT-based fast convolution in the second equation.

$$\begin{aligned}
W_n(s) &= \sum_{n'=0}^{N-1} x_{n'} \sqrt{\frac{\delta t}{s}} \Psi_0 * \left[\frac{(n'-n)\delta t}{s} \right] \\
W_n(s) &= FFT^{-1} \left[\sum_{k=0}^{N-1} \hat{x}_k \left(\sqrt{\frac{2\pi s}{\delta t}} \hat{\Psi}_0 * (s\omega_k) e^{i\omega_k n \delta t} \right) \right] \\
\hat{x}_k &= \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-2\pi i k n} \\
\omega_k &= if \left(k \leq \frac{N}{2}, \frac{2\pi k}{N\delta t}, -\frac{2\pi k}{N\delta t} \right)
\end{aligned}$$

In the above equations, the Ψ function is the mother wavelet, $*$ symbolizes a complex conjugation, N is the data series length, s is the wavelet scale, δt is the sampling interval, n is the localized time index, and ω is the angular frequency. Each of the equations contains a normalization so that the wavelet function contains unit energy at every scale.

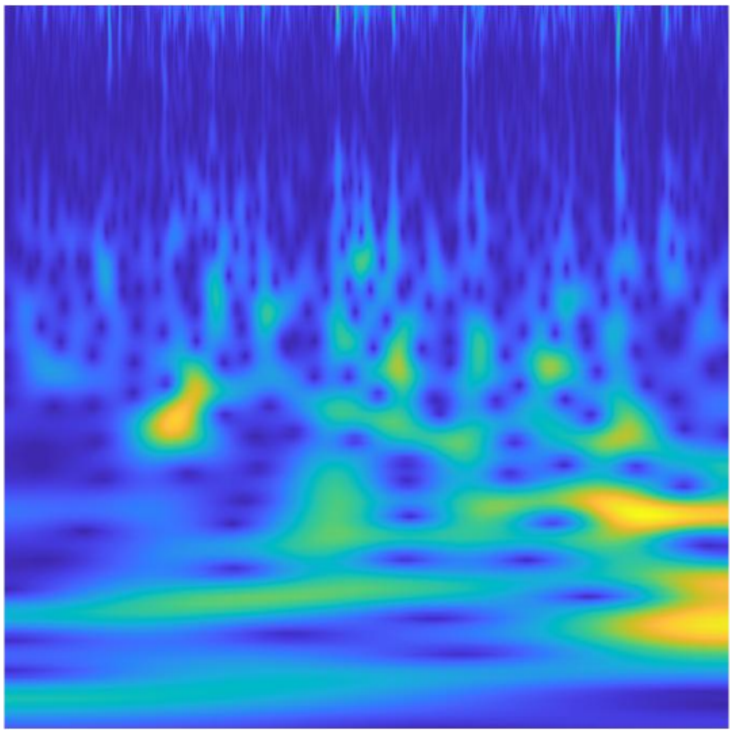


Figure 4.2: Example of a Continuous Wavelet Transformation image

Gramian Angular Field

Gramian Angular Field (GAF) [48] images represent a time series in a polar coordinate system instead of the typical Cartesian coordinates. In the Gramian matrix, each element is actually the cosine of the summation of angles. They preserve temporal dependency since time increases as the position moves from top-left to bottom-right.

The following formula is relative to the Gramian Angular Difference Field:

$$\begin{aligned} GADF &= [\cos \phi_i + \phi_j] \\ &= \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} \end{aligned}$$

While the one below here is relative to the Gramian Summation Field:

$$\begin{aligned} GASF &= [\sin \phi_i + \phi_j] \\ &= \sqrt{I - \tilde{X}^2}' \cdot \tilde{X} - \tilde{X}' \cdot \sqrt{I - \tilde{X}^2} \end{aligned}$$

$X = (x_1, x_2, ..., x_n)$ is a time series of n real-valued observations. $\tilde{X}$ is the rescaled X so that the values fall in the interval $[-1, 1]$ or $[0, 1]$. I is the unit row vector $[1, 1, ..., 1]$. Lastly ϕ:

$$\phi = \arccos(\tilde{x}_i), -1 \leq \tilde{x}_i \leq 1, \tilde{x}_i \in \tilde{X}$$

The respective images are Figure 4.3 and Figure 4.4.

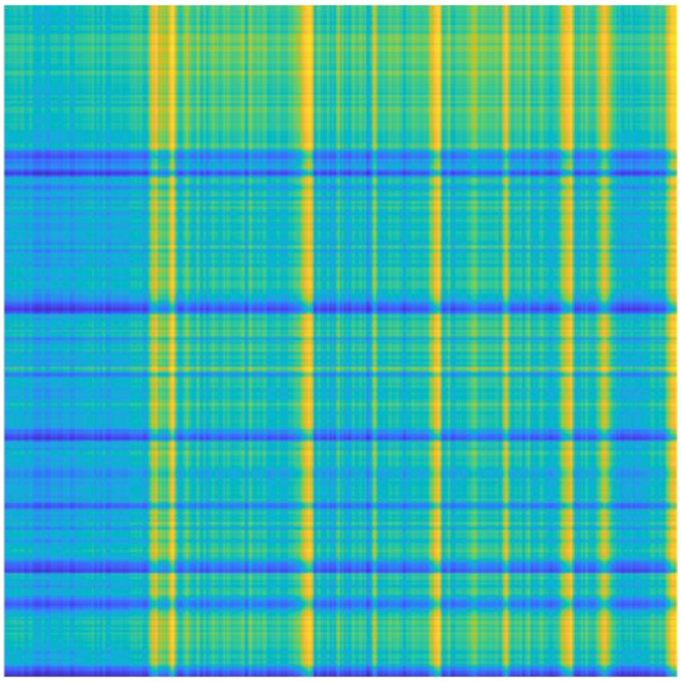


Figure 4.3: Example of a Gramian Angular Difference Field Image

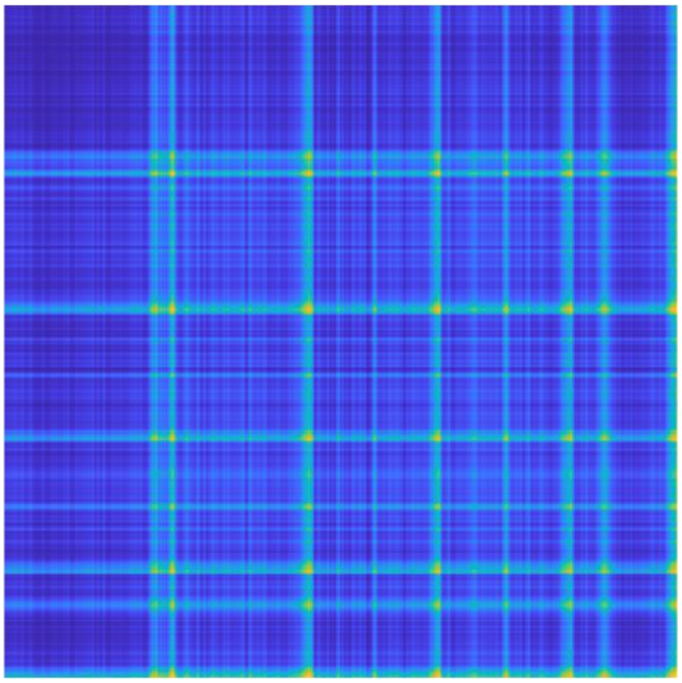


Figure 4.4: Example of a Gramian Angular Summation Field Image

Markov Transition Field

We used the Markov Transition Field (MTF) used in this paper by Wang et al. on generating images starting from time series to improve classification models.[48]

$$M = \begin{bmatrix} w_{ij|x_1 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_1 \in q_i, x_n \in q_j} \\ w_{ij|x_2 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_2 \in q_i, x_n \in q_j} \\ \vdots & \ddots & \vdots \\ w_{ij|x_n \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_n \in q_i, x_n \in q_j} \end{bmatrix}$$

M is constructed as follow: Given a time series $X = (x_1, x_2, ..., x_n)$, a data point x_u is assigned to its corresponding quantile bins $q_j (1 \leq j \leq Q)$, where Q is the number of states. In this way we can construct from X a Markov chain, deriving the QxQ Markov transition matrix (W) where $w_{ij}(1 \leq i, j \leq Q)$ in W is the frequency with which a data point in the state q_j is followed by a data point in state q_i. After normalizating by $\sum_j w_{ij} = 1$, W is the Markov transition matrix, where w_{ij} represents now the transition probability of $q_i \rightarrow q_j$.Thus, M_{uv}) in the MTF denotes the transition probability of $q_i \rightarrow q_j$, where x_u is the data point at time step u and $x_u \in q_i$ and x_v is the data point at time step v and $x_v \in q_j$. By assigning the probability from the quantile at time step i to the quantile at time step j at each pixel Mij , the MTF M actually encodes the multi-span transition probabilities of the time series. The main diagonal M_{ii} captures the probability from each quantile to itself (the self-transition probability) at time step i.

The example of an MTF image can be seen in Figure 4.5.

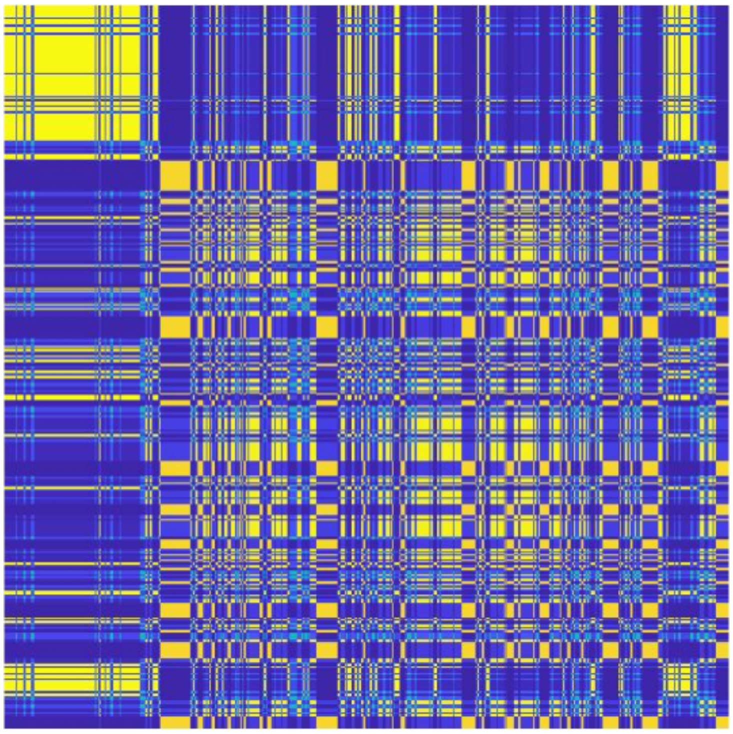


Figure 4.5: Example of a Markov Transition Field Image

Self-Similarity Matrix

The self-similarity matrix (S) is a graphical representation of pairwise similarities of sequences in a data series.

To explain the similarity in our Self-Similarity Matrix, we used spatial distance.

$$S(i) = |x(i) - x|$$

In the formula i is the index of the column of the image x. The image extracted is Figure 4.6.

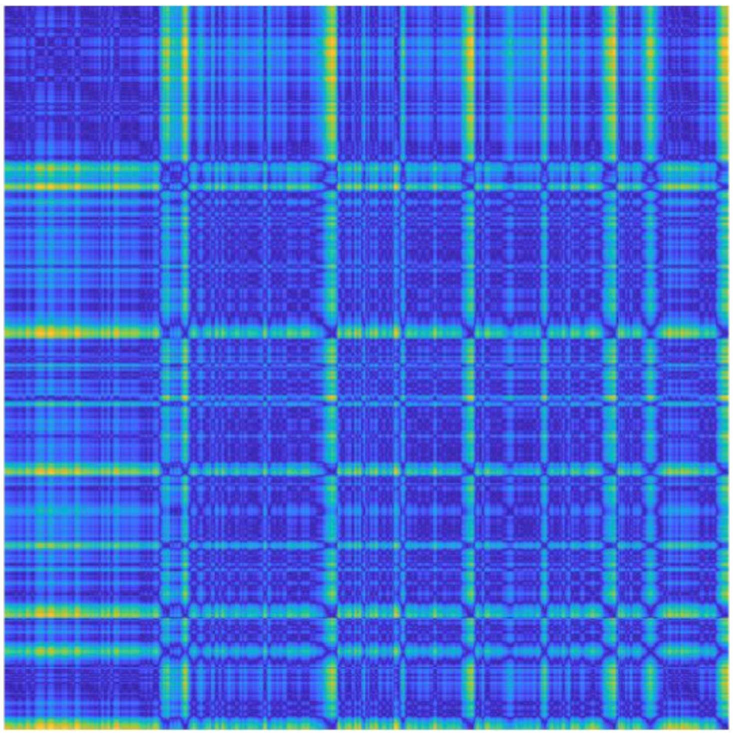


Figure 4.6: Example of a Self Similarity Matrix Image

Recurrence Plot

The Recurrence Plot (RP) is a plot that shows, for each time step i, the frequency that a phase space trajectory visits approximately the same area in the phase space at time step j.

$$R(i,j) = \begin{cases} 1 & \text{if } \|\vec{x}(i) - \vec{x}(j)\| \leq \epsilon, \\ 0 & \text{otherwise} \end{cases}$$

The figure for the Recurrence Plot is Figure 4.7.

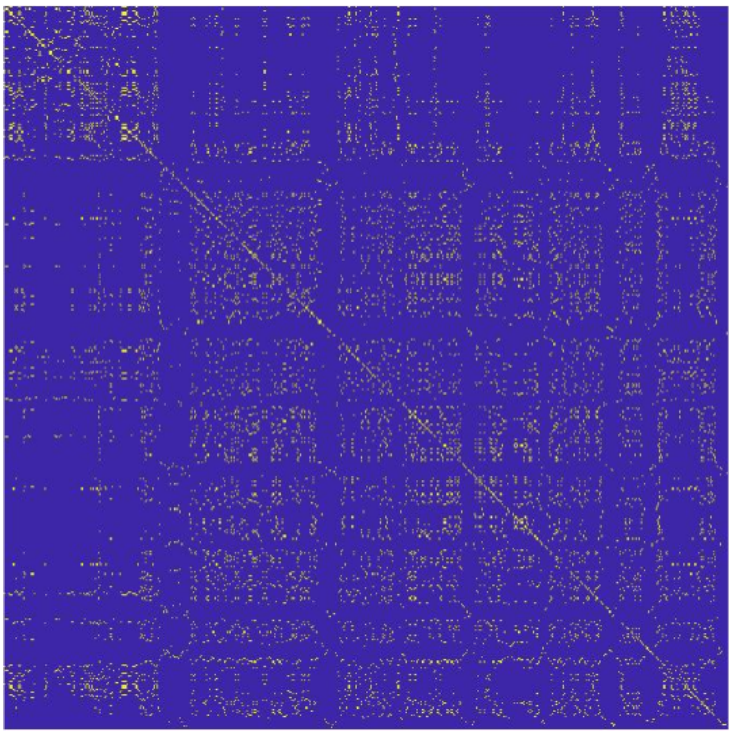


Figure 4.7: Example of a Recurrence Plot Image

Power Spectrogram

A power spectrogram (PS) is a visual representation of the spectrum of a signal's frequencies as it varies with time. The most common way to show spectrogram is using a heat map. It uses a system of color-coding to represent different intensity values.

From a time series, we can create spectrograms in two ways: approximated as a filterbank that results from a series of band-pass filters, or a more modern way that uses the Fourier transform on the time series. Even though these methods differ in their representations, they are equivalent to a certain extent. The former is analog processing of the signal, while the latter is a digital process thanks to the FFT.

The data is broken up into chunks, which usually overlap, and Fourier transformed to calculate the magnitude of the frequency spectrum for each chunk. Each vertical line in the image corresponds to a chunk, a measurement of magnitude versus frequency for a specific moment in time. These so-called spectrums are then put sequentially to form the image.

To retrieve the image then we need to apply a short-time Fourier transform (STFT) on the signal $s(t)$ and window width ω:

$$\text{spectrogram}(t, \omega) = |\text{STFT}(t, \omega)|^2$$

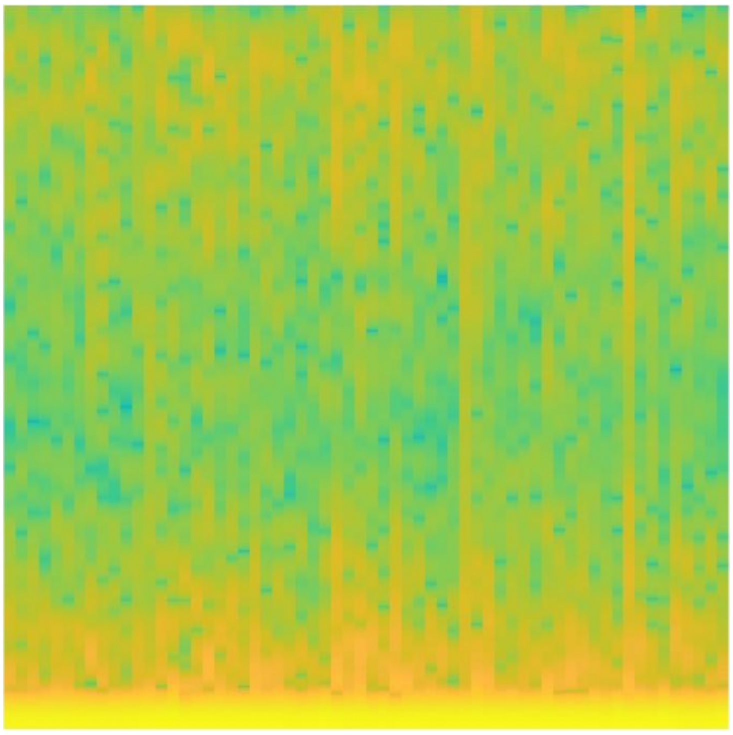


Figure 4.8: Example of a Power Spectrum Image

Persistence Spectrum

The persistence spectrum (PSP) is a histogram in power-frequency space. The brighter or "hotter" color in the image means how long a particular frequency persists in a signal as the signal progresses.

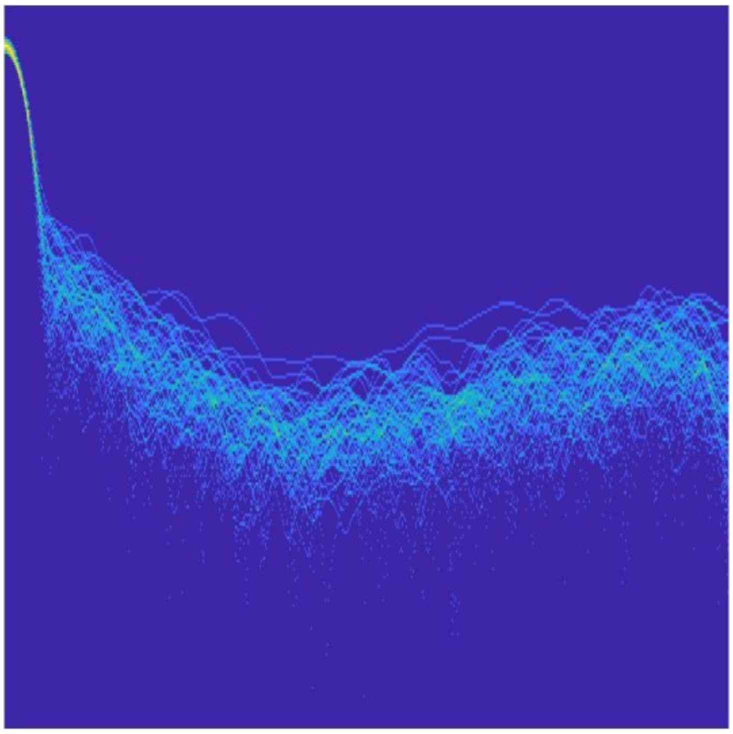


Figure 4.9: Example of a Persistence Spectrum Image

4.3.2 Our Model

Our CNN model was developed using a basic configuration inspired by the LeNet architecture. It possesses the basic units of a CNN: convolutional layer, pooling layer, and fully connected layers. Although in our model, we introduced also Batch Normalization and Dropout to penalize our model.

The input of our model is a tensor that is created by concatenating all tensors derived by the images previously introduced. We used only a tensor as it has to contains multiple images transformation of the same signal. The labels have been set as 0 for healthy and 1 for pathological.

The sizes of the images, as stated in chapter 2, varies from 32x32x3 to 256x256x3. After some hyper-parameter tuning, we found that the best kernels and filters were respectively: 5x5 and 16/32.

Model structure:

1. **Input layer**: A tensor made by concatenating 8 images
2. **Convolutional 2D Layer**: 16 filters, 5x5 kernel, activation function ReLU and no padding
3. **Batch Normalization Layer**
4. **Max Pooling 2D Layer**: pool size 2x2
5. **Convolutional 2D Layer**: 32 filters, 5x5 kernel, activation function ReLU and no padding
6. **Batch Normalization Layer**
7. **Max Pooling 2D Layer**: pool size 2x2
8. **Dropout**: with 80% drop out rate
9. **Flatten Layer**: to unroll the output of the convolutional layers in order to be fed to the fully connected layer
10. **Dense Layer**: 64 neurons and activation function ReLU
11. **Batch Normalization Layer**
12. **Dropout Layer**: with 80% drop out rate
13. **Dense Layer**: 16 neurons and activation function ReLU
14. **Output Layer**: 2 neurons with activation function Softmax for classification

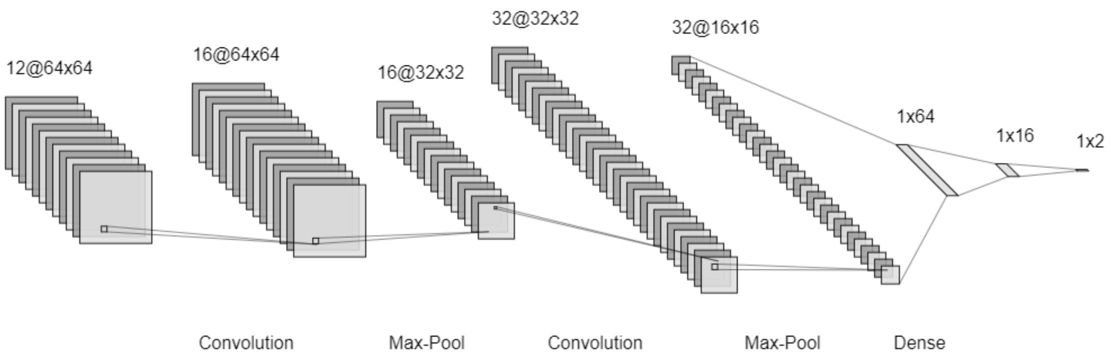


Figure 4.10: Our CNN model

4.3.3 Results Achieved

The following results are only for the images of size 32x32x3 and 64x64x3, as the main limitation of our work was that by using Kaggle, we have a limited amount of RAM, and working with images bigger than 64x64 results impossible.With the 32x32x3 images, we have achieved an accuracy of 64%, while with the images 64x64x3, we reached a 68% accuracy. These results are congruent to the fact that the model should be able to learn better in discriminating classes by having more information.

Lastly, we want to point out that we trained this model using different combinations of images to understand which were the more meaningful. In table 4.1 we show the top-performing combinations. These results will be later used for our mixed approach. Indeed, this model performs worse than the MLP, but we understood that the model could learn from these types of transformed images.

4.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of artificial neural networks where the fully connected network can be seen as a directed graph on a temporal sequence. In this way, we achieve a temporal dynamic behavior. They use their internal state (memory) to process data, and thanks to the loops in their networks, they allow information to persist. They found application in handwriting recognition, speech recognition, temporal predictions, music composition, and other domains.

The main differences in RNNs are that there are finite and infinite ones. For the former, the network is a directed acyclic graph that can be "unrolled" and replaced with a feedforward neural network, while the latter is a directed cyclic graph that can not be unrolled.

One citation from T. Siegelmann that summarizes the power of the RNNs is as follow:

> *"With enough neurons and time, RNNs can compute anything that can be computed by a computer"* [49]

One main limitation of the RNNs is the problem of the vanishing or exploding gradient. When we have the vanishing gradient, it is due because the further it goes through the network, the lower the gradient is and the harder it is to train the weights. The same applies to the exploding gradients, but the weights tend to grow exponentially.

Even though in our work we did not use RNNs, we had to introduce them as LSTM are a special kind of RNN, but they are better in remembering information for long periods of time and have been used by us.

4.4.1 Long Short-Term Memory

LSTMs [50] have a much more complicated structure compared to RNNs, but they perform better in terms of memory as they are capable of remembering for a longer time. Since their introduction, they have been highly refined, and now they work exceptionally well on a large variety of problems and are now widely used.

LSTMs have been designed to avoid the long-term dependency problem. They are similar to RNNs since they are a chain of repeating modules of neural networks.

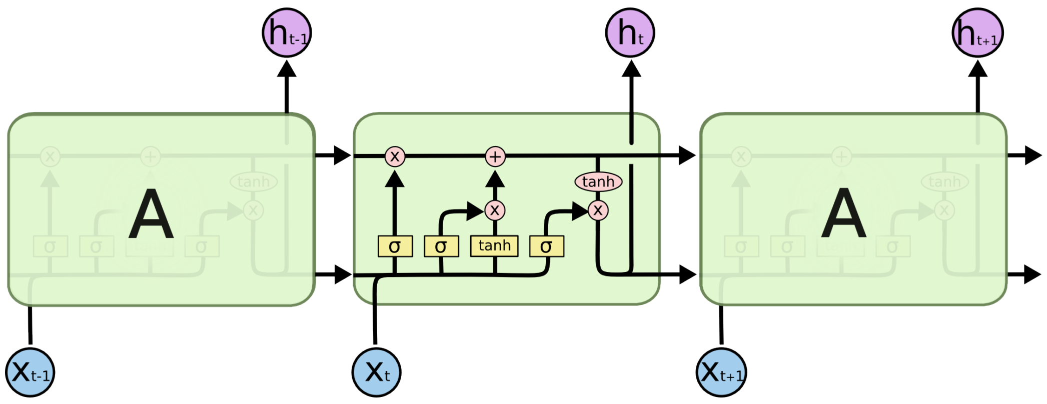


Figure 4.11: LSTM architecture

The RNNs usually have a simple tanh layer, while LSTMs keep the chain structure, but the repeating module is not comprised of only a single layer, but there are four. Let us introduce the core element of the LSTMs that is the cell state, which is the horizontal line running at the top of the figure 4.11. [51]

It conveys the information through the entire chain and can have small interactions, and sometimes the information may remain unchanged. This information can be regulated thanks to the other layers that we talked about earlier, and these are called gates:

1. **Input Gate**: It is where information flows into the cell state, which, thanks to the sigmoid layer it decides which values will be updated, while the tanh layer creates new possible values that might be added to the state.

2. **Forget Gate**: It decides what has to be forgotten by the cell state. The sigmoid layer takes the decision, returning a value between 1 and 0, which respectively means remember or forget this.

3. **Memory Gate**: It is responsible for updating the old cell state into a new one. By combining the output of the forget gate and the input gate, LSTMs forget the things that have been already decided while adding the data that is needed.

4. **Output Gate**: It outputs based on the cell state, but it is filtered thanks to a sigmoid layer that determines which part of the state we are going to output. Lastly, a tanh layer that filters out the parts that matter.

4.4.2 Our Model

Before explaining the model we used, we need to understand the data given as input to the model.

We have decided to use the FHR signals that were extracted from our dataset. As stated in chapter 2, we have taken only the signals with 2400 data points, and if there were more, we have used an overlapping technique to create multiple signals. We created a tensor that has been scaled using the MinMax function, which has been fed to our network.

The model structure is the following:

1. **LSTM Layer**: 8 units
2. **Batch Normalization Layer**
3. **LSTM Layer**: 8 units
4. **Batch Normalization Layer**
5. **LSTM Layer**: 8 units
6. **Output Layer**: 2 neurons for classification, using as activation function softmax

4.4.3 Results Achieved

The results achieved were not as we expected. We have fined tuned some hyperparameters, but the networks were not capable of learning. We reached a 55% accuracy, which indicates that our model was almost as good as random guessing.

We think that this scarce results is due to the nature of the signal. FHR signal is a stochastic signal compared to an almost periodic heartbeat signal. Thus, the LSTM struggles in finding patterns inside the signal.

Chapter 5 explains this limitation further and how we think we can try to improve the network and the results.

4.5 Mixed Approach

In this section, we will elaborate and explain our approach and the models created, and at the end of this section, we will explain the results achieved with all the networks.

Before beginning, it is essential to explain that we used a mixed approach to try and reach a higher accuracy. This decision was taken based on the assumption that deep learning networks can find connections between different types of inputs. Lastly, we want to highlight that we have used the same output layer for all the combinations of networks that we have used.

4.5.1 MLP & CNN

Our first idea was to create a network where the output of the MLP and CNN was concatenated. In order to do so, the last output layer has to be taken out since the classification is done once the data from the two networks have been combined. The networks are the same as explained previously. We are going to refer to their whole model structure as MLP and CNN for convenience. As input for this network, we have used the same dataset already fed to the MLP and the images used to train our CNN model earlier.

The final model is the following:

1. **MLP Network**
2. **CNN Network**
3. **Concatenation**: This is where the output from both networks gets concatenated into a single input that is fed to subsequent fully connected layer
4. **Dense Layer**: It has 128 neurons, with activation function ReLU, it is needed in order to teach the network how to deal with the outputs
5. **Output Layer**: 2 neurons for classification and as activation function softmax

4.5.2 MLP & LSTM

Another idea was to use the good accuracy achieved by our MLP to boost the results achieved from the LSTM network. Hence the two outputs have been concatenated as we did for the mixed MLP and CNN network. In this case, since we already named the MLP, we will refer to the LSTM as the whole network structure, with the last classification layer taken out. This model input was the dataset used for the MLP and the raw signals used for the LSTM.

The structure of this network is as follow:

1. **MLP Network**
2. **LSTM Network**
3. **Concatenation**
4. **Dense Layer**: 128 neurons and activation function ReLU
5. **Output Layer**: 2 neurons for classification and as activation function softmax

4.5.3 MLP & CNN & LSTM

The last idea was to combine the MLP, CNN, and LSTM to try and see if this would perform any better than any of the other models that we have tried.

This network instead combined all the inputs already used for the networks specified.

Below here you can find our last structure:

1. **MLP Network**
2. **LSTM Network**
3. **Concatenation**
4. **Dense Layer**: 128 neurons and activation function ReLU
5. **Output Layer**: 2 neurons for classification and as activation function softmax

4.5.4 Final Results

As we expected, mixed networks, in some instances, performed better than every single model. Earlier, we said that we had tried multiple combinations of images to check, which could lead to better results. When we trained these mixed models, we used the top-performing sets of images that can be found in table 4.1. In the following pages, there are the results of these combinations for all the explained models.

Overall the best performing model is the CNN and MLP combined together as it reached an accuracy of 79.1%, as shown in table 4.2. This is an improvement from our high coming from the MLP of 75.7%. This model was able to use the high accuracy of the MLP to influence and boost the accuracy of the CNN on the images. We reached a peak of 85.2% in our training session. The second-best performing model is the MLP, CNN, and LSTM combined, which reached almost identical results as the CNN and MLP, thus showing that the LSTM is not providing additional information; actually, it slightly penalized the model. For reference check table 4.3. The model with the least accuracy reached was the MLP and LSTM together. Even though the MLP was the only single model to perform better, the LSTM did not add any information. In fact, the accuracy remained the same as for the MLP 75.7%. By looking at the results, we can have a clear view on which images have been the most impactful on our model. The first one is the GADF as it has been chosen always as important. Then we have the PS and PSP which have been chosen multiple times. The least important are GASF and RP. A summary of the results from all the models used can be found in table 4.4.

CWT	***GADF***	***GASF***	***MTF***	***RP***	***S***	***PS***	***PSP***	***Accuracy***
0	1	0	0	0	0	1	1	68.1%
1	1	1	1	1	0	1	1	67.7%
1	1	1	1	1	1	1	0	67.5%
0	1	0	0	0	0	0	1	67%
1	1	0	1	0	1	1	0	66.9%
1	1	1	1	1	0	1	0	66.4%
0	1	0	0	0	1	1	0	66.3%
0	1	0	0	0	1	0	1	66.3%
0	1	0	0	0	0	0	1	66.2%
1	1	0	1	0	1	0	1	65.9%

Table 4.1: Top 10 best performing combinations of images for the CNN

CWT	***GADF***	***GASF***	***MTF***	***RP***	***S***	***PS***	***PSP***	***Accuracy***
0	1	0	0	0	0	1	1	79.1%
1	1	1	1	1	0	1	1	78.6%
1	1	1	1	1	1	1	0	78.4%
0	1	0	0	0	0	0	1	78.4%
1	1	0	1	0	1	1	0	78.4%
1	1	1	1	1	0	1	0	78.3%
0	1	0	0	0	1	1	0	78.3%
0	1	0	0	0	1	0	1	77.9%
0	1	0	0	0	0	0	1	77.9%
1	1	0	1	0	1	0	1	77.8%

Table 4.2: Top 10 best performing combinations of images for the mixed model (CNN+MLP)

CWT	***GADF***	***GASF***	***MTF***	***RP***	***S***	***PS***	***PSP***	***Accuracy***
0	1	0	0	0	0	1	1	78.9%
1	1	1	1	1	0	1	1	78.6%
1	1	1	1	1	1	1	0	78.2%
0	1	0	0	0	0	0	1	78%
1	1	0	1	0	1	1	0	77.8%
1	1	1	1	1	0	1	0	77.8%
0	1	0	0	0	1	1	0	77.5%
0	1	0	0	0	1	0	1	77.1%
0	1	0	0	0	0	0	1	76.8%
1	1	0	1	0	1	0	1	76.6%

Table 4.3: Top 10 best performing combinations of images for the mixed model (CNN+MLP+LSTM)

Model	***Accuracy***
MLP	75.7%
CNN	68.1%
LSTM	55%
MLP+CNN	79.1%
MLP+LSTM	75.7%
MLP+CNN+LSTM	78.9%

Table 4.4: Best results of the models tested

Chapter 5

Conclusions

In this work, we presented a novel way to discriminate between pathological pregnancies and not. We took advantage of Machine Learning approaches, with a specific interest in Deep Learning. The Machine Learning approach focused on analyzing numerical data of the mother and the CTG's signal. While the Deep Learning approach, thanks to its flexibility and versatility, enabled us to perform wider tests including images, signals and raw data.

5.1 Machine Learning versus Deep Learning

We have seen how for Machine learning one of the best performing model was the eXtreme Gradient Boosting (XGB). It was chosen four out of seven times for each dataset selected by the features selection algorithms, as can be seen in table 3.3. Although our highest R-Squared was only of 0.7.

On the other hand, neural networks, especially the hybrid approach of MLP and CNN combined, reached an accuracy of 79.1% with a peak value of

85.2%. NNs have proven to be better and more reliable for our problem, as they tend to generalize better and understand intricate connections between features in the dataset through a hierarchical learning process. We think this results is especially due to the power of deep learning and the amount of data provided. As can be seen in figure deep learning algorithms are known to perform better compared to old machine learning algorithms when the the amount of data provided is significant. [52]

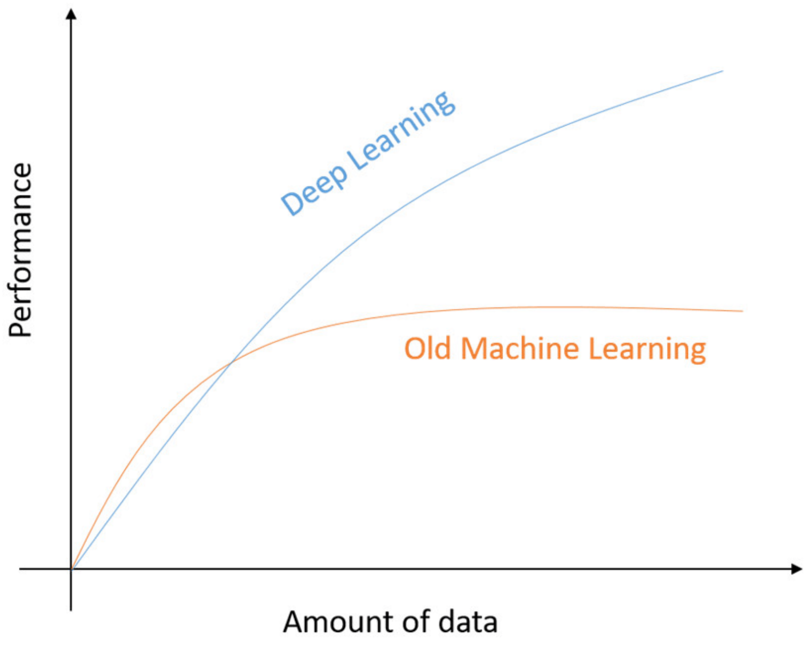


Figure 5.1: Performance of Deep Learning and Machine Learning with respect to the amount of data

5.2 Limitations

While making our research we noticed some main limitations:

1. **Kaggle environment and its memory usage**: As stated in chapter 4, we were not able to train our model on images bigger than 64x64. The limited resolution might reduce the ability of the model to learn and improve.

2. **FHR signal and LSTM training**: Due to its stochastic nature we think that the network finds it difficult to learn patterns in the signal. Especially because some signals have been corrected, hence some points were interpolated and there might be some noise that could impair the ability of model.

3. **Dataset and mother's information**: Our dataset did not contain any data pertaining to the mother's general status like: BMI, blood pressure, smoking status and so on. We think that this type of information will greatly improve our models.

5.3 Future Work

We have multiple proposal to improve our work:

1. **Image Resolution**: Using images of size 128x128 or 256x256 would be highly beneficial for our model.

2. **LSTM training**: Signal's raw data might need to be associated with other temporal aspect of the signal, like mean values of some particular features from our dataset.

3. **Different images transformation**: Other images transformations exist and they should be explored to understand their relevance for model training.

4. **Website and user interface**: Implement a website where doctors could upload their raw signals and patient's data such that our algorithms can extract all the features and evaluate the state of the pregnancy based on our pre-trained deep learning model.

Bibliography

[1] John Hopkins Medicine. 4 common pregnancy complications, 2018.

[2] G. Piana A. Bonalumi P. Brambilla D. Arduini, G. Rizzo and C. Romanini. Computerized analysis of fetal heart rate: I. description of the system (2ctg). page 159, 1993.

[3] Cerutti S Arduini D Signorini MG, Magenes G. Linear and nonlinear parameters for the analysis of fetal heart rate signal from cardiotocographic recordings. pages 50(3):365–74, 2003.

[4] Chandraharan E Ayres-de Campos D, Spong CY. Intrapartum fetal monitoring expert consensus panel. figo consensus guidelines on intrapartum fetal monitoring: Cardiotocography. pages 13–24, 2015.

[5] Wood C Walker DW. Temperature relationship of the mother and fetus during labor. 322(10):107(1):83–7.), 1970.

[6] Ververs IAP et al Mantel R, Van Geijn HP. Automated analysis of antepartum fetal heart rate in relation to fetal rest-activity states: a longitudinal study of uncomplicated pregnancies using the sonicaid system 8000. pages 71: 41–51., 1996.

[7] Arthur M Searle N Metheny WP Ruedrich DA Castillo RA1, Devoe LD. Athe preterm nonstress test: effects of gestational age and length of study. pages 160(1):172–5, 1989.

[8] Sorokin Y Rosen MG Dierker LJ Jr, Pillay SK. Active and quiet periods in the preterm and term fetus. pages 60(1):65–70, 1982.

[9] Dunn LJ Jordaan HV Segreti A Krebs HB, Petres RE. Intrapartum fetal heart rate monitoring. i. classification and prognosis of fetal heart rate patterns. pages 133(7):762–72, 1979.

[10] Cibils LA. Clinical significance of fetal heart rate patterns during labor. ii. late decelerations. pages 123(5):473–94, 1975.

[11] Parer JT Harris JL, Krueger TR. Mechanisms of late decelerations of the fetal heart rate during hypoxia. pages 144(5):491–6, 1982.

[12] Demirci F Ozden S. Significance for fetal outcome of poor prognostic features in fetal heart rate traces with variable decelerations. pages 262(3–4):141–9, 1999.

[13] Acién P Tortosa MN. Evaluation of variable decelerations of fetal heart rate with the deceleration index: influence of associated abnormal parameters and their relation to the state and evolution of the newborn. pages 34(3):235–45, 1990.

[14] Johnson N McNamara H. The effect of uterine contractions on fetal oxygen saturation. pages 102(8):644–7, 1995.

[15] Zhang Y Zhang Y Zhang X Shao L Zhao Z, Deng Y. Deepfhr: intelligent prediction of fetal acidemia using fetal heart rate signals based on convolutional neural network. *BMC Med Inform Decis Mak*, 19:286, 12 2019.

[16] Alessio Petrozziello, Ivan Jordanov, A.T. Papageorghiou, Christopher Redman, and Antoniya Georgieva. Deep learning for continuous electronic fetal monitoring in labor. 07 2018.

[17] P. Fergus, Abir Hussain, Dhiya Al-Jumeily Obe, De-Shuang Huang, and Nizar Bouguila. Classification of caesarean section and normal vaginal deliveries using foetal heart rate signals and advanced machine learning algorithms. *BioMedical Engineering OnLine*, 16, 07 2017.

[18] ms Iraji. Prediction of fetal state from the cardiotocogram recordings using neural network models. *Artificial Intelligence in Medicine*, 96, 03 2019.

[19] Magenes G Signorini MG, Fanelli A. Monitoring fetal heart rate during pregnancy: contributions from advanced signal processing and wearable technology. 2014.

[20] Caron FJM Swartijes JM van Worden EE Jongsma HW Mantel R, Van Geijn HP. Computer analysis of antepartum fetal heart rate: I. baseline determination. pages 25: 262–72, 1990.

[21] G. S. Dawes K. J. Dalton and J. E. Patrick. Diurnal, respiratory and other rhythms of fetal heart rate in lambs. 127:414, 1977.

[22] A. A. Baschat. Neurodevelopment following fetal growth restriction and its relationship with antepartum parameters of placental dysfunction. vol. 37(no. 5):pp. 501–514, 2011.

[23] Bellver J Redman CW Serra V, Moulden M. The value of the short-term fetal heart rate variation for timing the delivery of growth-retarded fetuses. (115):1101–1107, 2008.

[24] Ruozi-Berretta A et al Anceschi MM, Piazze JJ. Validity of short term variation (stv) in detection of fetal acidemia. (31):231–236, 2003.

[25] B. Versteeg A. F. L. Veth L. A. M. Stolte J. Janssens J. De Haan, J. H. Vam Bemmel and T. K. A. B.Eskes. Quantitative evaluation of fetal heart rate. i. processing methods. (3):95, 1971.

[26] A. Forsythe S. Yeh and E. H. Hon. Quantification of fetal heart rate beat-to-beat interval differences. (41):355, 1973.

[27] Zhou S Tang D Wang C Wu G Li X, Zheng D. Approximate entropy of fetal heart rate variability as a predictor of fetal distress in women at termpregnancy. (84):837–843, 2005.

[28] S. M. Pincus. Approximate entropy (apen) as complexity measure. (5(1)):110–117, 1995.

[29] Michael Lecocke and Ken Hess. An empirical study of univariate and genetic algorithm-based feature selection in binary classification with microarray data. page 313–27, 2007.

[30] Y. W. Chen X. Zeng and C. Tao. Feature selection using recursive feature elimination for handwritten digit recognition. page 1205–1208, 2009.

[31] Miron Kursa and Witold Rudnicki. The all relevant feature selection using random forest. 06 2011.

[32] Krzysztof Grabczewski and Norbert Jankowski. Feature selection with decision tree criterion. page 6 pp., 12 2005.

[33] Robert Tibshirani. Regression shrinkage and selection via the lasso. page 58, 1996.

[34] Cadima J Jolliffe IT. Principal component analysis: a review and recent developments. page 374, 2016.

[35] James Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 03 2012.

[36] Kuk Lee Joanne Peng and Gary Ingersoll. An introduction to logistic regression analysis and reporting. *Journal of Educational Research - J EDUC RES 96*, 13:3–14, 2002.

[37] David Cutler Adele Cutler and John Stevens. Random forests. *Machine Learning - ML*, 45:157–176, 01 2011.

[38] Ran Chu, Wei Chen, Guangmin Song, Shu Yao, Lin Xie, Li Song, Yue Zhang, Lijun Chen, Xiangli Zhang, Yuyan Ma, Xia Luo, Yuan Liu, Ping Sun, Shuquan Zhang, Yan Fang, Taotao Dong, Qing Zhang, Jin

Peng, Lu Zhang, Yuan Wei, Wenxia Zhang, Xuantao Su, Xu Qiao, Kun Song, Xingsheng Yang, and Beihua Kong. Predicting the risk of adverse events in pregnant women with congenital heart disease. *Journal of the American Heart Association*, 9(14), 2020.

[39] Alaa Tharwat. Adaboost classifier: an overview. 02 2018.

[40] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boostingsystem. page 785–794, 2016.

[41] Y. Bengio and Olivier Delalleau. On the expressive power of deep architectures. page 1, 10 2011.

[42] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[43] M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal*, 37:233–243, 1991.

[44] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 07–09 Jul 2015.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[46] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel

Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. 08 2016.

[47] Alexandre Grossmann, Richard Kronland-Martinet, and J. Morlet. Reading and understanding continuous wavelet transforms. *In Wavelets: Time-Frequency Methods and Phase Space*, page 2, 01 1989.

[48] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. 05 2015.

[49] H.T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Progress in Theoretical Computer Science. Birkhäuser Boston, 2012.

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[51] Andrej Karpathy. Understanding lstm networks, 2015.

[52] Md. Zahangir Alom, Tarek Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Nasrin, Mahmudul Hasan, Brian Essen, Abdul Awwal, and Vijayan Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8:292, 03 2019.