Politecnico di Torino

Master Degree in Biomedical Engineering

Advanced Imputation Techniques for Missing Values in Weaning-Focused Datasets Derived from MIMIC-IV

Supervisor
Prof.ssa. Gabriella Balestra

Candidate
Claudia Gambara s292201

Co-Supervisors
Ing. Andrea Scotto
Prof.ssa Samanta Rosati

Graduation Session: October 2024

Abstract

Missing values are a significant challenge in the application of Artificial Intelligence (AI), particularly in healthcare. Incomplete datasets can lead to biased models and unreliable predictions, making it essential to develop robust methods for data imputation. Traditional approaches, such as replacing missing values with the mean or median, and more advanced statistical methods, have been widely applied. However, recent advancements in Machine Learning (ML) and Deep Learning (DL) offer promising alternatives.

In healthcare, time is a crucial factor, especially in Intensive Care Units (ICUs) where timely decisions can impact patient outcomes. Health-related datasets often incorporate time information, which can be leveraged to improve the accuracy of imputation methods. The correlation between variables and their temporal patterns provides a valuable resource for reconstructing missing values in time-series data.

This thesis focuses on developing and testing datasets for ICU patients undergoing weaning from mechanical ventilation, using data derived from MIMIC-IV. The datasets include physiological variables that are useful for assessing a patient's readiness to initiate the weaning process. Weaning is a crucial phase in critical care, requiring precise timing to avoid complications associated with prolonged mechanical ventilation, which is linked to higher morbidity and mortality. Studies confirm that the timing of weaning significantly affects the success patient outcomes.

To address the issue of missing data, this work employs state-of-the-art deep learning models, specifically the Bidirectional Recurrent Imputation for Time Series (BRITS) network, designed to handle time-series data with missing values. BRITS utilizes both forward and backward sequences to capture temporal dependencies, making it well-suited for imputation in healthcare settings. Additionally, BRITS has been compared with the Multidirectional Recurrent Neural Network (MRNN), another advanced technique for handling sequential data with missing values. The comparison focuses on evaluating the performance of each model using metrics such as Mean Absolute Error (MAE) and Mean Relative Error (MRE) calculated on different percentages of missing values.

The ultimate goal of this research is to create a fully imputed, accurate dataset that can assist clinicians in making informed decisions about weaning. By comparing the performance of BRITS and MRNN, this work aims to identify the most effective approach for handling missing data in critical care. Furthermore, the methodology developed can be adapted for other time-sensitive clinical applications, highlighting the broader potential impact of machine learning-based imputation in healthcare.

Contents

Chapter 1

Introduction

In recent years, Artificial Intelligence (AI) has become an important tool in healthcare. It is utilized in diagnostics, treatment planning, and patient care. Machine and Deep Learning algorithms have demonstrated their effectiveness in making predictions, identifying patterns and providing personalized recommendations. However, one of the main problems of these algorithms are the input data, as achieving high performance critically depends on having access to large datasets that are:

- **Informative**: the data must represent the problem that is trying to solve in order to effectively train the algorithm.
- **Relevant**: even if, some variables are congruent with the problem, it is possible for multiple variables to provide the same information, this can cause redundancy. To avoid this, it is useful to select only the relevant variables.
- **Consistent**: the information must be consistent across all records; inconsistencies can lead to confusion and consequent poor performances.
- **Complete**: the presence of missing values can disrupt model training and cause poor performance. Furthermore, according to multiple studies a high rate of missing values may impact conclusion in primary care studies [1].

This work aims to restore the completeness of the data, as incomplete datasets are a common occurrence in real-world scenarios. In healthcare it can be due to partially filled-out surveys or, for example, incomplete medical records [1].
While this work main focus is on useful data for the weaning process, the algorithms that have been used can be applied to different datasets. The decision to concentrate on weaning comes from the need for new innovative methods that can help clinicians in the choice of the optimal time for starting the weaning process [20].
The next chapter will provide an in-depth look on why the time is crucial in the weaning process and why new solutions are needed.

Chapter 2

State of the art

2.1 Missing Values imputation methods

Because of the importance of missing values in datasets, in recent years, managing them has become a significant challenge in data analysis and Machine/Deep Learning. To determine the most appropriate method to apply in each case a key start-point is the classification of missing data in three categories [2]:

1. **MCAR (Missing Completely at Random)**: the absence is completely unrelated to the data; so, the fact that a value is missing does not depend on any other variables, or even on the value itself. For example, if we imagine asking people to answer a survey about their holidays and we observe some missing information that is randomly distributed among the data.

2. **MAR (Missing at Random)**: in this case the absence is related to other variables but it is not related to the value itself. For example, about the same survey, we notice that people aged 50-65 do not respond to some of the questions.

3. **MNAR (Missing Not at Random)**: in this case the absence is related to the value that is missing. For example, in the same survey we notice that people tend to avoid responding to a specific question, because of the question itself.

It is important to correctly classify the missing values, in fact while MCAR are rare, not paying attention to MNAR may lead to a not representative dataset because of the lack of data from a possible key subgroup of the dataset[2]. Imputing missing values have become important mostly for MNAR. In fact, this type of missing is called "not ignorable", unlike MCAR and MAR which are ignorable because they cannot bias the dataset. Moreover, missing data have an important impact on time series data, that not only have to take into account the cross-relation between variables, but also the relationship with time. This section will present a brief summary of the methods used to impute missing values in various fields, including healthcare.

2.1.1 Simple methods

- **LOCF (Last Observation Carried Forward) and NOCB (Next Observation Carried Backward)**: in the first case the missing value is filled as the last non missing observation in the time series, while in the second case, the missing value is substituted with the subsequent observed value. The problem with these imputation methods is that they can introduce a bias if the data are not stationary [3].

- **Median/Mean/Mode**: in this case the missing data are replaced with median if the data are ordinal, with mean if the data is continuous or mode if the data is categorical. With these methods, the problem may be the variance of the data that is not been taken into account[3].

- **Linear/Spline Interpolation**: in the first case the missing values are estimated by fitting a line between the preceding and subsequent data; the method is appropriate in case of linear relation among observations. In the second case, a polynomial interpolation is used, allowing a representation for non-linear data, but it is assumed smoothness in the data [3].

2.1.2 Time Series specific methods

- **Kalman Filtering**: it is a statistical algorithm useful for handling noisy data that operates recursively. Based on Durbin and Koopman (2012) [5] there is an observed sequence which may contain missing values and an unobserved sequence. The first is related to the second, while the second evolves following a structure. The filter predicts the missing value based on past values and the system's known dynamics. Once the prediction is done, the filter adjusts its imputation based on the difference between the prediction and the real value [4]. The basic filter can work only with linear data, but there are variants of it that can work with other types of data. However, it is always necessary to make a model of the system in order to know the system's dynamics.

- **Moving Average Imputation**: this uses the average of nearby values to impute the one missing.

- **ARIMA (Autoregressive Integrated Moving Average)**: it predicts future values using statistical techniques. It performs well if the past data points are not below a certain minimum, as the prediction relies on the past values that must have been recorded in regular intervals. The method has three components: the autoregressive terms, the difference in information and the size of the moving average window [6]. However, in healthcare this model is less utilized than in economics because it may struggle with irregular patterns, which are commons in this scope.

2.1.3 Probabilistic and Statistical methods

- **Expectation – Maximization (EM)**: it performs maximum likelihood estimation based on a full set of variables. It imputes the value and then optimizes the model; these two steps are repeated until convergence is reached. This algorithm is mostly utilized in case of density estimation, such as in clustering algorithms. However, this algorithm makes the strong assumption that all the variables relevant to the problem are known, which is not always the case [7].

- **Multiple Imputation by Chained Equations (MICE)**: this method operates through steps, starting with a simple imputation of the missing value (using the mean, for example) to fill all the missing in the data, then only one variable is reset as missing. This variable is now assumed as the dependent variable from all the other ones that are assumed independent, at this point the real imputation is performed using regression models. Once the variable has its value, it is used as independent when imputing the other variables that are missing [8]. All these steps are performed in cycle, more than once, in order to adjust the imputation. Even if MICE can be utilized with a lot of missing data, this method assumes that the missing data are MAR types, so using it when the type is not the one expected can introduce biases in the imputation.

- **Gaussian Process (GP)**: it is a non-parametric probabilistic model that takes into account uncertainty given by the noise of the data that is assumed being on some underlying function [9]. GP models the data as a distribution over all the possible functions, providing a probabilistic estimate of the function that fits the data. Then it predicts the missing values using the seen data and the kernel function, also providing the uncertainty through the variance. The classic method requires the inversion of a covariance matrix of the given points. Clearly, this is computationally expensive for a large set of points, however there are some new methods that try to solve this issue.

2.1.4 Machine and Deep Learning methods

Machine Learning (ML) is a subset of Artificial Intelligence. Classic Machine Learning depends on human intervention, in fact usually pre-processing on data is needed in order to allow to a computer system patterns identification and learning.
A subset of ML is Deep Learning, the main difference between them is how their algorithm learns and how much data is needed [23]. In fact, while Deep Learning automates much of the feature extraction process, it also needs larger datasets in comparison to ML. The way the algorithms learns, both in Machine Learning and Deep Learning can be supervised, when labeled data are provided, or unsupervised when there are no labels.
Artificial Neural Networks are a subset of ML and the backbone of Deep Learning. They are made of node layers: input layer, one or more hidden layers and output layer. Each of this node, is an artificial neuron that connects to the next in order to pass information. The information can be passed forward, or forward and backward between layers. Usually we define "Deep Learning" an Artificial Neural Network that has more than three layers including input and output [23].

- **k-Nearest Neighbors (kNN) Imputation**: it is a valid method to impute missing values especially for numerical variables. The idea behind kNN is to cluster all similar data in various clusters, this way missing values are imputed relying on similar known values. To establish the similarity a distance is calculated, typically the Euclidean distance for numerical values and Manhattan distance for categorical ones [10], the data, then, are ranked based on the selected metric. Once the k neighbors, with k as a numerical parameter that must be set, are identified, the missing value is imputed using the mean of the identified closer points if the dare are numerical, or the mode otherwise [10]. Since k represents the number of "clusters" used to establish similarity, it is a crucial parameter. In fact, low k might lead to high variance, while large k might cause to lose individual patterns. This problem can be limited by applying this method recursively, starting with a low k increasing the value to obtain the optimal result. The advantage is the simplicity of this method, while the disadvantages are that for large datasets it can be computationally expensive, moreover as the number of features increases the distance between points becomes less meaningful badly affecting its performances.

- **Random Forest**: it is primarily used for classification/regression tasks, but it is also a valid method to impute missing values. It consists of multiple decision trees: a decision tree is a supervised machine learning algorithm, it splits datasets into smaller subsets based on the feature, it has nodes and leaves, exactly like a tree. The first node is the "root node" which is where the first decision is made. Once the decision has been determined, the splitting continues in internal nodes that represent other decisions, in the end there are leaf nodes, these makes the classification/regression. In

order to proceed with the splitting, the algorithm uses a specific criterion to separate data, for classification tasks Gini Impurity or Information Gain are often the choices, while for regression tasks the goal is often to minimize Mean Squared Error or Variance. Once the tree is build it is used following the path from root node to leaf node; this is applied recursively to the subsets in order to obtain the desired outcome: a label for classification or a value for regression. When multiple trees are used simultaneously, their outcome can be combined, this guarantees better performances because it is possible to combine multiple results in order to choose the one that fit better the problem. When this method is used to impute missing values, the algorithm uses the pattern that has been created using the known values to make predictions. While Random Forest is powerful to make imputations it is also computationally expensive, especially for large datasets.

- **RNN (Recurrent Neural Network)**: is a special type of Artificial Neural Network (ANN) that is mostly used with time series or sequences of data. Its strength consists in the fact that it can maintain a "memory" across time steps. To better understand RNN it can be useful a brief introduction to feed forward neural networks. In a feed-forward neural network the information moves only in one direction: from the input layer to the output one, through the hidden layers. In contrast to RNN, these kinds of neural network do not have memory. In RNN the information cycles [11] so during the decision making counts not only the current input but also the hidden state from the previous time step. A visual representation of the differences between these two networks can be seen in Fig. 2.1.

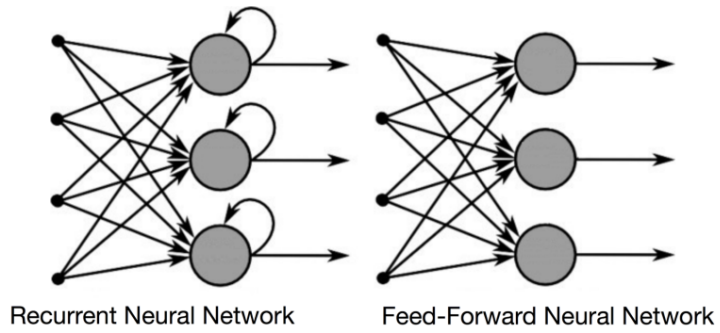


Figure 2.1: Differences between FNN and RNN. (Picture taken from Ref. [11])

In RNN to arrive to the final output, each neuron computes a sum of inputs and applies an activation function (most commons are sigmoid, linear, ReLu[11]). Once the output is produced there is a loss function that measures the differences between the output produced and the expected outcome (ground truth). At this point backpropagation is used to compute

gradients of the loss function, in conclusion this is used to adjust the weights of neurons according to the error each one produced. Following gradient computation, an optimizer such as Adam can be used to minimize the loss function. However, while the feedback loop is what makes RNN powerful, it introduces also some challenges, as it propagates, the gradient may become too small, making difficult for the model to learn long-term dependencies. This is the reason why some variants of RNN have been created such as: LSTM (Long Term Short Memory) and GRU (Gated Recurrent Unit).

- **LSTM (Long Short-Term Memory)**: it is a variant of RNN that can work with long-term dependencies[14]. LSTM has three types of gates 1) input gate: decides which information to store in the memory cell to do so it opens when the input is important while it closes when it is not. 2) Forget Gate: this is trained to decide which information, that successfully made into the memory cell, is still important and which can be discarded. The mechanism for taking the decision is the same as the input gate: the gate opens or closes basing on the importance of the information. 3) Output Gate: is responsible of which information should be considered to generate an output. Even this gate has the opening and closing mechanism. In this network cell state (or memory cell) and hidden state are updated separately, because the cell state has long term memory while the hidden state has short term memory. The complexity of this model makes it perfect for long and complex time series.

- **GRU (Gated Recurrent Unit)**: this is a variant of RNN, created to capture long-term dependencies without the complexity of LSTM [12], because the mechanism through it handles memory cell state is simpler than in LSTM. The architecture has: an input vector that takes in input a sequence. Followed by the hidden layer where the recurrent computation is done, it represents the network's memory of previous inputs. Then, there is the reset gate, it takes in the current input and the previous hidden state, then generates a binary vector to determine which hidden states to reset. In order to arrive to the update gate, it is necessary the candidate activation vector which is a modified version of the previous state that has been changed by the reset gate. To obtain the vector this new version of the hidden state is combined with the current input. Afterwards, this vector is given in input to the update gate that decides how much of the vector fitting into the new hidden state. This is done by creating a new binary vector that controls how much of the candidate activation vector can be incorporated into the new hidden state. This newly created hidden state is given to the output layer that produces the output. In particular, GRU-D is the variant of GRU created to handle the missing values. The key differences between the two is that GRU-D introduces decay mechanisms for missing data, meaning that if an observation is missing GRU-D uses last available value, however if the value is missing for a long time, the network gradually reduces its confidence in the old value. The decay is also applied

to the hidden state, this allows network's memory to fade if there has been a long gap since last observation.

2.1.4.1 NAOMI: Non-AutOregressive Multiresolution Imputation

NAOMI is a method first published in 2019, by Liu et al[16]. It is a novel network to impute long range sequences given arbitrary missing patterns; the main goal is to fix the problem of having missing values for a long time.

- **Key aspects**
  - Non Autoregressive Modeling: the majority of traditional deep generative model for time-series imputation, model the value at the current time step based on the previous time steps, leading to errors over long sequences. NAOMI uses a different algorithm and makes the predictions non sequentially.
  - Generative Adversial Training: normal Generative Adversial Networks uses the discriminator for the entire sequences, given the length, it may ignore sequential dependencies. NAOMI incorporates adversial training through Generative Adversarial Imitation Learning (GAIL) that uses a discriminator to compare the generate sequences with ground-truth ones.
  - Multiresolution Generation: it operates on multiple resolutions, so it does not make one-time step prediction at a time. This allows to consider coarse-grained and fine-grained dynamics.

- **Architecture**
  NAOMI has two components 1) a bidirectional encoder that converts the incomplete sequences into hidden states [16] 2) a multiresolution decoder that fills in the missing values based on the hidden states [16]. The way the bidirectional encoder maps the incomplete sequence gives to the model a time complexity similar to the one of autoregressive models. Decoder imputes missing values recursively from coarse to fine-grained resolutions. It starts with larger intervals and progressively fills in smaller gaps. This approach is called "Divide and Conquer".
  In Fig. 2.2 it is shown a sum up of NAOMI architecture.

- **Dataset**
  This network has not been tested on healthcare datasets by the authors. Instead, it has been tested on billiard and basketball trajectories.
  - Billiard trajectories: the dataset contains 4000 training and 1000 test sequences, each with 200-time steps (more than what can usually be seen in healthcare data). In each sequence, 180 to 195 time steps are masked, leaving only a few observed time-steps. The trajectories are initiated with random positions and velocities that is maintained

constant, also friction is not taken into account. This dataset is used to test the performance of NAOMI with deterministic dynamics.

– Basketball player movement: this dataset contains the movement trajectories of professional basketball players, tracking their (x,y) coordinates. The players are 5, sampled at 6.25 Hz over a sequence of 50 time steps. The dataset has 107146 training elements and 13845 test. Each trajectory has 40 to 49 missing values. This dataset is used to test NAOMI with stochastic dynamics.

Although, this method is efficient with long sequences of missing data, it is computationally expensive, moreover it needs ground-truth values during the training. In conclusion, NAOMI is optimal for long starting sequences that contain many missing values.

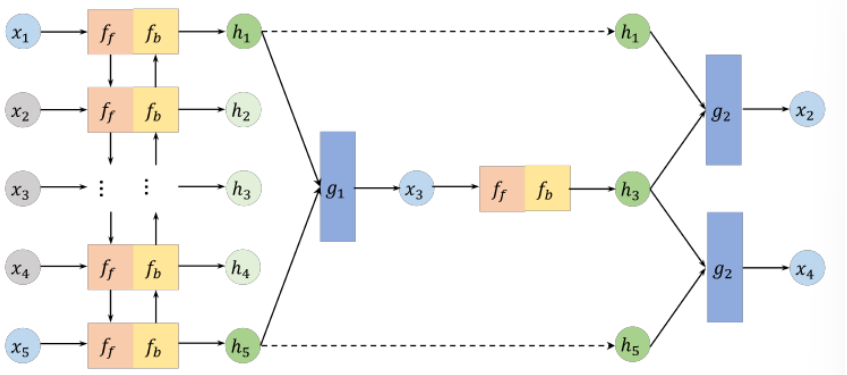


Figure 2.2: NAOMI Architecture to impute a sequence of five, x is the input while h are the hidden states. (Picture taken from Ref. [16])

2.1.4.2 TAME: Time-Aware Multi-Modal Auto-Encoder

TAME is a method to impute missing values in healthcare datasets published in 2020 by Zhang et al[17]. It provides an innovative solution to handle missing values considering muti-modal EHR data. While it is particularly specialized for sepsis patients, it lays the foundation for a more personalized medicine.

- **Key aspects**
  - The model incorporates data from multiple sources such as demographics, diagnoses, lab tests, and medications, it then uses a Bidirectional Long Short-Term Memory network to model time series.
  - It takes into consideration the fact that often data comes from different sources so time gaps may be irregular. It emphasizes recent observations, making the model sensitive to the timing of past observations. This is the Time-Aware Mechanism.
  - After imputing missing data, the model uses Dynamic Time Warping (DWT) to measure the similarity between patients. They are then clustered into subphenotypes using a weighted k-means algorithm.

- **Architecture**

  - Multimodal embedding: 1) Embedding for demographics, diagnoses, and medications: during this step different types of data are being represented in a common numerical format that can be processed by the network. 2) Embedding for variable values: each variable is mapped into a vector. This process translates discrete values into representations that capture their semantic characteristics. 3) Time embedding: the time gaps between observations are incorporated using time embeddings.

  - Max-Pooling Layer: The various embeddings produced are combined through a max-pooling layer (meaning the maximum value in a group is selected to reduce dimensionality and simultaneously maintain information). This generates fixed-size vectors that represents relevant information from different inputs.

  - BiLSTM: The fixed vectors are then given in input to Bidirectional Long Short-Term Memory network.

  - Time-Aware Attention Module: this mechanism assigns more weight to relevant information helping model focus on the most important aspects of time series during the imputation process. This mechanism improves accuracy of imputations.

  - Imputation of missing values: Output from BiLSTM and Time-Aware Attention Module is used to impute missing values.
    A schematic view of the architecture of TAME is shown in Fig. 2.3

- **Dataset**
  This network has been tested on two different healthcare datasets by the authors.

  - MIMIC-III: Medical information Mart for Intensive Care database combines health data from more than 40000 patients admitted to intensive care unit (ICU) of the Beth Israel Deaconess Medical Center between 2001 and 2012. For this study a subset of 11715 patients with sepsis was selected. The variables selected are 27 and are all related to better understand severity of sepsis. Moreover, this dataset contains a lot of missing values and irregular sampling intervals.

  - DACMI: it consists of clinical data collected from 8267 patients, focusing on 13 lab tests that are measured irregularly overtime. The missing rate here is lower and can go from 1% to 15% for the different clinical variables.

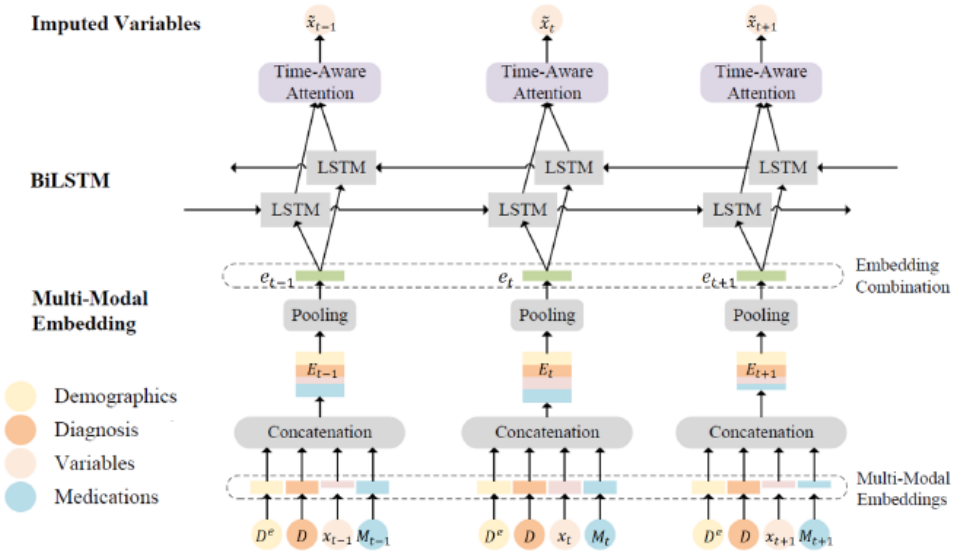


Figure 2.3: A schematic view of TAME architecture (Picture taken from Ref. [17])

2.1.4.3 CATSI: Context-Aware Time Series Imputation

CATSI is a method used to handle missing data in clinical time series, published in 2019 by Yin et al[18]. This network aims to improve accuracy by unifying the context (temporal and variable relationship) within the time series data.

- **Key aspects**
  - Context Awareness: it considers both temporal dependency and inter-variable relationships. Because of clinical data generally follows temporal patterns, this methods aim to find it in order to make better imputations. Moreover, CATSI recognizes that some variable are interdependent, these correlations are used to provide context for the imputations.
  - Handling irregular sampling and missing patterns: clinical time series often have irregular sampling and missing data, CATSI is designed to handle those aspects.
  - Multimodal Data Imputation: it balances intra and inter variables relationships in order to have more robust imputations.
- **Architecture**
  - Context-Aware Recurrent: for the creation of the context vector two approaches are used: 1) summation of basic statistics: a multilayer perceptron (MLP) is used to approximate the function that consider mean, standard deviation, missing rate and length of time series for each patient. 2) RNN-based encoder is used to capture the complex

temporal dynamics. The entire time series is given to the Gated Recurrent Unit encoder, the hidden states at the final steps are then taken as the context vector, that summarizes the entire sequence. The outputs from the two approaches are then concatenated.

- Cross-Feature Component: the main idea here is the correlations between different features to estimate the missing values. There are two steps 1) linear transformation calculation: a linear transformation is applied to the vector of observed features generating an intermediate representation. 2) nonlinear function application: the intermediate representation is then passed through a nonlinear function approximated by MLP to generate the imputed value.
- Combination of the imputations: once both the imputations have been obtained they are combined.
  A scheme of the architecture of CATSI is showed in (Fig. 2.4)

- **Dataset**
  This method has been tried on DACMI dataset which has been previously discussed.

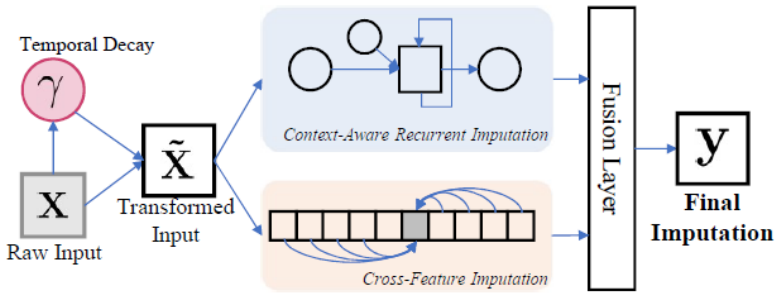


Figure 2.4: Framework of CATSI. (Picture taken from Ref. [18])

2.1.4.4 GP-VAE: Gaussian Process - Variational Autoencoder

GP-VAE is a deep-learning based method for imputing missing data in multivariate time series published in 2019 by Fortuin et al[13]. The proposed model combines a variational autoencoder (VAE) with a Gaussian Process (GP) to achieve dimensionality reduction and data imputation while handling missing values.

- **Key aspects**

  - Imputation for Multivariate Time Series: imputing missing values using temporal correlations intra and inter variables.

  - Deep Learning and Gaussian Process: Deep Variational Autoencoders are used to perform dimensionality reduction, while Gaussian Process is used to capture temporal dynamics in a latent space without missing data.
  - Uncertainty Estimation: it provides a probabilistic framework that offers interpretable uncertainty estimation.

- **Architecture**
  A VAE is a generative model that learns latent representations of complex data. It consist of an encoder that maps the data to a latent distribution and a decoder that reconstructs the original data from the latent representation.

  - Encoding: an incomplete time series is passed through the encoder (CNN) in order to obtain latent representation (mean and variance).
  - Sampling: sampling is applied to latent distributions to obtain latent vectors.
  - GP modeling: Gaussian Process is used to model latent vectors. This, allows to impute missing values utilizing temporal dependencies captured by GP kernel.
  - Latent imputation: missing latent values are imputed using GP.
  - Decoding: imputed latent vectors are passed through decoder (MLP) to reconstruct complete time series.

- **Dataset**
  To test this network Physionet Challenge 2012 dataset has been used. It includes multivariate time series irregularly sampled from ICU patients. There are 37 variables, that have been recorded in the first 48 hours since the patient has been admitted in the ICU. Moreover, not all the variables are numerical, some of them are categorical. All the patients have been divided in two sets: 4000 in training set and 4000 in test set.

2.1.4.5 BRITS: Bidirectional Recurrent Imputation for Time Series

BRITS is one versatile method that has been proposed in 2018 by Cao et al[15]. It is one of the most important deep-learning methods to impute missing values in time series in healthcare datasets, in fact, it is often used as a comparison for novel methods. It will be discussed in detail in Chapter 3; however it is now presented a brief summary.

- **Key aspects**

  - Bidirectional RNN: it permits to improve the process of imputing missing data by using past and future data.
  - Self-Learning imputations: introduces a self-refining process where imputed values are improved by the model through the process.

  - No pre-imputation required: there is no need to prepare data to imputation filling missing values with a simple imputation method.
  - Imputation/Classification: it performs classification simultaneously with the imputation process; this allows the model to improve both tasks.

- **Architecture**

  - Complement vector: considering there is no initialization of missing values, a new complementary vector is needed, this vector contains the previous hidden state.
  - History-based estimation: thanks to the complement vector it is possible to make a history based estimation, however this alone do not take into consideration the fact that variables can be correlated.
  - Feature-based estimation: due to the reason previously explained, a feature based estimation is also made. This estimation considers the correlation among variables.
  - Combined vector: this vector combines the two estimation previously made: 1) History based and 2) Feature based.
  - Combination of forward and backward: the imputation is made for the backward direction and for the forward one. The two imputations are then combined in order to obtain the final result.

- **Dataset**
  To test this network Physionet Challenge 2012 dataset has been used. As it has been previously discussed it has categorical and numerical variables, in this study the categorical are left out. The final dataset has up 78% of missing values.

2.2 Ventilator Weaning

Weaning is a technique widely used in Intensive Care Unit (ICU). Its use, started around 1950[19], and from its start, the procedure has been significantly improved. According to "Weaning from Mechanical Ventilation" by Guilherme Sant'Anna and Martin Keszler: "weaning from mechanical ventilation is the process of decreasing the amount of ventilatory support". This process can be classified in:

- **Simple:** high probability of success (30-60%) and lower mortality (5-10%)[19]. In this case is important to correctly determine readiness for weaning as promptly as possible.
- **Difficult:** in this case a patient requires up to three Spontaneous Breathing Trial (SBT) or as long as seven days to achieve weaning. In this case the priority is to identify reversible factors that can contribute to the failure of SBT. The success rate in this case is (15-40%) while the mortality is (10-30%)[19].
- **Prolonged:** in this case the SBT either fail or the patient requires more than seven days to be successfully weaned from mechanical ventilation. The percentages of success and mortality for this category are the same as the "difficult" one.

The weaning process follows a two-step approach, the first one is for readiness assessment while the other is for SBT. Criteria that must be evaluated are both objective and clinical; the evaluation begins as early as 48 to 72 hours after ICU admission. Clinical issues such as electrolyte imbalances, fluid overload and pneumonia must be addressed before the objective criteria are applied[19].
Time in this procedure is a crucial aspect. That is because prolonged use of mechanical ventilation increases risk of complication such as tracheal injuries and musculoskeletal deconditioning and other health problems [22], moreover, delays in weaning process are linked to higher rates of morbidity and mortality. Also, weaning phase accounts around 42% of the time that patients spend on mechanical ventilation[22].
Despite the existence of a protocol to follow in order to establish optimal time to start the weaning process, some issues have not been addressed. In fact, the failure rate still ranges from 10 to 15% of ICU patients in United States [20]. Moreover, a study conducted on 17 trials with 2434 patients[21], with the goal of finding differences in the outcome of the weaning procedure with and without protocol, found that the total geometric mean duration of mechanical ventilation in the protocolized weaning group was on average reduced only by 26% showing the needs for new methods to predict the optimal weaning time.

Chapter 3

Methodology

3.1 Initial Dataset

3.1.1 Introduction to MIMIC-IV

Nowadays, digital health is an expanding market, in fact, in 2022 the global digital health market was valued approximately 220 billion US dollars[24] and its value is expected to increase. Despite this growth, and the adoption of electronic record systems, often archiving systems are not designed to permit the use of data in research.
The Intensive Care Unit, however, is a data-rich environment, due to the importance of immediate intervention and the severity of the situation. Mostly of the projects based on data of ICU patients are made on MIMIC, a database with demographics digitally transcribed for over 90 patients[25]. This was followed by MIMIC-II with a larger number of patients and more data, taken from digital systems directly. In 2015 MIMIC-III was published, it expanded MIMIC-II with over 40,000 patients[25]. Other datasets have been published over the years such as the HiRD database with 34,000 patients[25].
The dataset used in this work is based on MIMIC-IV, which is the result of a collaboration between Beth Israel Deaconess Medical Center (BIDMC) and Massachusetts Institute of Technology (MIT). This dataset covers admissions between 2008 and 2019, making it contemporary with relatively new information collected at BIDMC as part of routine clinical care and other activities such as monitoring. The patients were deidentified and transferred via VPN connection to MIT. In Fig. 3.1 an overview of the development process of MIMIC IV.

3.1.2 Weaning Focused - Dataset

In order to obtain the final dataset, a subset of MIMIC-IV focused on weaning a patient from mechanical ventilation is needed [1]. The starting point was selecting inclusion and exclusion criteria:

[1]This work has been conducted by Ing. Andrea Scotto

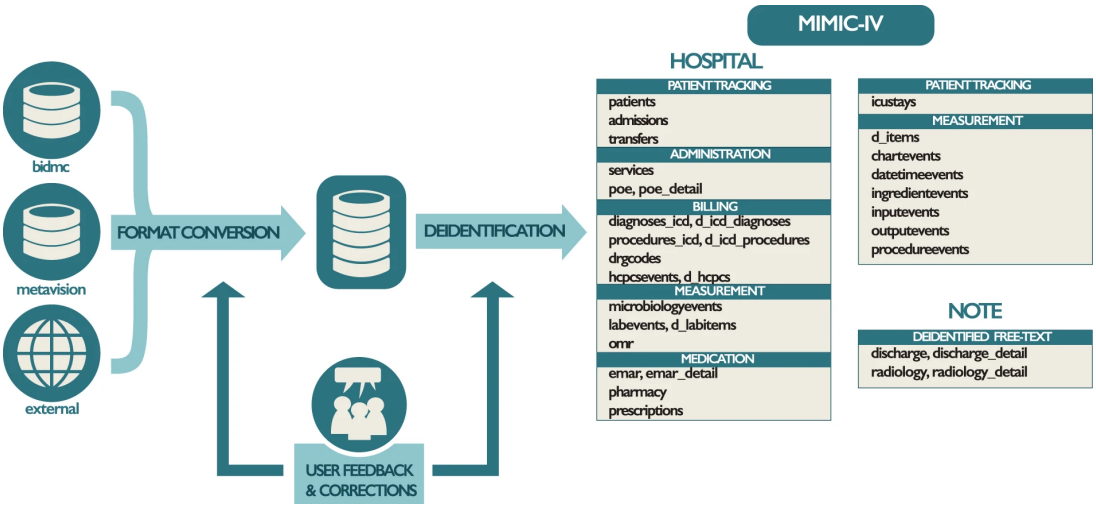


Figure 3.1: Information from data acquisition from BIDMC, MetaVision (the ICU information system) and external sources are merged via SQL. In the end, deidentification algorithm are applied. On right side of the figure an overview of the tables present in MIMIC-IV. From "MIMIC-IV, a freely accessible electronic health record dataset" [25].

- **Inclusion Criteria:** definition of events associated with patients undergoing invasive mechanical ventilation. A patient is considered intubated if they manifest parameters related to mechanical ventilation; the first occurrence of one of these parameters marks the beginning of the event, while its presence within the following 12 hours marks its continuation. From this step 38,756 events have been selected.
- **Exclusion criteria:** Events were excluded if any of the following occurred:
  - Subsequent ventilation events during the same ICU admission (11,399 events).
  - Ventilation events that occurs after the first ICU admission (4218 events).
  - Ventilation events with a duration of less than 24 hours (13,754).
  - Events associated to patients who died before the weaning was attempted (1256).

Once the preliminary dataset of intubated patients was created, a total of 8,129 patients were accepted. The following step was defining in which cases a patient can be considered extubated. In order to do this, some criteria were set:

- Absence of an event correlated to invasive ventilation which is defined by variables as "tracheotomy tube" or "endotracheal tube".
- Presence of an event correlated to non-invasive ventilation, defined by variables like "Bi-PAP mask" or "CPAP mask".

- Initiation of oxygen-therapy.

After the extubation, another classification based on its success is performed:

- **Extubation success:** if no parameters related to invasive ventilation are observed in the 48 hours following the extubation (6069 patients out 8129).
- **Extubation failure:** if there are parameters correlated to invasive ventilation in the 48 hours following extubation because the patient has been reintubated (1597 out of 8129), or if the patient dies within 48 hours post-extubation (463 patients out of 8129).

Once the patients were selected their data have been divided in multiple subsets.

3.1.2.1 Blood Test - Subset

The blood test subset contains 24 variables, shown in Table 3.1 for 8115 patients, meaning not all the selected patients have blood test analysis.
The records are 2,522,035 and the table is structured as follows:

- *Subject ID*: an eight digits number to identify the patients in a deidentified way.
- *Hadm ID:* an integer number identifying a specific admission to the hospital. It is possible for a patient to have multiple hadm ID.
- *Item ID:* an integer that identify uniquely a variable.
- *Chart time:* it is a date with year, month, day, hour and minutes of the record, every record has its own time of registration. The years have been modified to protect patient privacy.
- *Value:* a numerical value assumed by the variable.
- *Range Lower/Upper:* represent the minimum and maximum for a variable.
- *Value Unit of Measurement:* it is a string with the unit of measurement for the variable considered.
- *Parametro:* it is a string with the name of the variable.

3.1.2.2 Blood Gas Test - Subset

The blood gas test subset contains 20 variables, shown in Table 3.2 for 8129 patients, meaning all the selected patients have at least one record in the blood gas analysis subset. The records are 3,617,926 and the table is structured as the blood test subset, however, in this subset there is no the information on the Upper/Lower Range.

Table 3.1: Table with name and unit of measurement of all the variables contained in subset: Blood Tests

Variable with Unit of Measurement	**Variable with Unit of Measurement**
White Blood Cells (WBC) (k/uL)	Blood Urea Nitrogen (BUN) (mg/dL)
Creatinine (mg/dL)	Chloride, Whole Blood-Blood-Blood Gas (mEq/L)
Chloride-Blood-Chemistry	Hemoglobin-Blood-Blood Gas (g/dL)
Hemoglobin-Blood-Hematology (g/dL)	Glucose-Blood-Blood Gas (mg/dL)
Glucose–Blood-Chemistry (mg/dL)	Potassium, Whole Blood–Blood-Blood Gas (mEq/L)
Sodium, Whole Blood–Blood-Blood Gas (mEq/L)	Sodium–Blood-Chemistry (mEq/L)
International Normalized Ratio (INR)	C-Reactive Protein (CRP) (mg/dL)
C-Reactive Protein (CRP) (mg/dL)	Total Protein (g/dL)
Prothrombin Time (s)	Partial Thromboplastin Time (s)
Troponin (ng/L)	Alanine Aminotransferase (ALT/GPT) (U/L)
Albumin (g/dL)	Aspartate Aminotransferase (AST/GOT) (U/L)
B-type Natriuretic Peptide (BNP) (pg/mL)	Platelets (k/uL)
Bilirubin (mg/dL)	

Table 3.2: Table with name and unit of measurement of all the variables contained in subset: Blood Gas Tests

Variable with Unit of Measurement	**Variable with Unit of Measurement**
Free calcium (mEq/L)	Lactic Acid (mmol/L)
Total calcium (mEq/L)	Oxygen Saturation (%)
Calculated bicarbonate, whole blood (mEq/L)	Bicarbonate (mEq/L)
Base Excess (mmol/L)	pH
pCO2 (mmHg)	Arterial CO2 pressure (mmHg)
Venous CO2 pressure (mmHg)	pO2 (mmHg)
Arterial O2 pressure (mmHg)	Venous O2 pressure (mmHg)
O2 saturation by pulse oximetry (%)	SpO2 Desaturation Limit (%)
Central Venous O2 Saturation (%)	Arterial O2 Saturation (%)
ScvO2 Central Venous O2 Saturation (%)	ScvO2 (Presep) (%)

3.1.2.3 Vital Signs - Subset

The Vital Signs subset contains 25 variables, shown in Table 3.3 for 8129 patients, meaning all the selected patients have at least one vital sign record. The records are 10,396,973 and the table is structured as the blood test subset, except for Upper/Lower Range, that are not present in this subset. Instead, this subset has a *Stay ID* associated with the specific ICU admission.

Table 3.3: Table with name and unit of measurement of all the variables contained in subset: Vital Signs

Variable with Unit of Measurement	**Variable with Unit of Measurement**
Heart Rate (bpm)	Arterial Blood Pressure Systolic (mmHg)
Arterial Blood Pressure Diastolic (mmHg)	Arterial Blood Pressure Mean (mmHg)
Pulmonary Artery Pressure Systolic (mmHg)	Pulmonary Artery Pressure Diastolic (mmHg)
Pulmonary Artery Pressure Mean (mmHg)	Non-invasive Blood Pressure Systolic (mmHg)
Non-invasive Blood Pressure Mean / Diastolic (mmHg)	Respiratory Rate (insp/min)
Temperature (°C)	Manual Blood Pressure Systolic Left (mmHg)
IABP (Intra-Aortic Balloon Pump) Mean (mmHg)	Manual Blood Pressure Diastolic Left (mmHg)
Set Respiratory Rate (insp/min)	Spontaneous Respiratory Rate (insp/min)
Total Respiratory Rate (insp/min)	Arterial Blood Pressure (ART) Systolic (mmHg)
Aortic Pressure Signal Diastolic (mmHg)	Aortic Pressure Signal Systolic (mmHg)
Arterial Blood Pressure (ART) Diastolic (mmHg)	Arterial Blood Pressure (ART) Mean (mmHg)
Manual Blood Pressure Diastolic Right (mmHg)	Manual Blood Pressure Systolic Right (mmHg)
Temperature (°C from °F)	

3.1.2.4 Ventilation - Subset

The Ventilation subset contains 13 variables, shown in Table 3.4 for 8128 patients, meaning all the selected patients but one have at least one ventilation correlated record. The records are 2,470,421 and the table is structured as the vital signs subset.

Table 3.4: Table with name and unit of measurement of all the variables contained in subset: Ventilation

Variable with Unit of Measurement	**Variable with Unit of Measurement**
Positive End Expiratory Pressure (PEEP) set (cmH2O)	Fraction of Inspired Oxygen (FiO2) (%)
Tidal Volume set (ml)	Tidal Volume observed (ml)
Tidal Volume spontaneous (ml)	Minute Volume (L/min)
Positive Inspiratory Pressure (PIP) (cmH2O)	Plateau Pressure (cmH2O)
Mean Airway Pressure (cmH2O)	Positive End Expiratory Pressure (PEEP) total (cmH2O)
Transpulmonary Pressure (Insp. Hold) (cmH2O)	Pinsp (Draeger only) (cmH2O)
Rapid Shallow Breathing Index (RSBI) (ins/min/L)	

3.1.2.5 Drugs - Subset

The Drugs subset contains 23 ItemID but the variables are 21, because two of them have two codes for the same variable (Table 3.5). The patients contained in the dataset are 8065 while the total number of records is 836,511. This subset has a slightly different structure compared to the others; in fact, it does not have a time of registration for the record, but a time to indicate the start and one to indicate the end of the administration of the drug.

Table 3.5: Table with name and unit of measurement of all the variables contained in subset: Drugs

Variable with Unit of Measurement	**Variable with Unit of Measurement**
Epinephrine (mg)	Lorazepam (Ativan) (mg)
Cisatracurium (mg)	Diazepam (Valium) (mg)
Dobutamine (mg)	Dopamine (mg)
Midazolam (Versed) (mg)	Fentanyl (mg)
Phenylephrine (mg)	Hydromorphone (Dilaudid) (mg)
Norepinephrine (mg)	Milrinone (mg)
Vecuronium (mg)	Propofol (mg)
Dexmedetomidine (Precedex) (mcg)	Morphine Sulfate (mg)
Fentanyl (Concentrate) (mg)	Meperidine (Demerol) (mg)
Methadone Hydrochloride (mg)	Phenylephrine (200/250) (mg)
Phenylephrine (50/250) (mg)	

3.1.2.6 Other relevant variables

In a task like defining the outcome of an intubated patient who has to overcome weaning, some other relevant variables are:

- Glasgow Coma Scale (GCS): it is a parameter ranging from 3 to 15, used to provide an objective measure of the level of impaired consciousness in all types of acute medical and trauma patients. The aspects taken into consideration are three: eye-opening, motor and verbal responses. The higher the value the better the health condition. It is a parameter that can be considered not static because it can be recalculated multiple times during the same ICU admission.

- Sequential Organ Failure Assessment (SOFA): it is a parameter that assess the performances of several organs and then assigns a score based on that. The higher this parameter the higher the probability of death. It is a particularly useful parameter in cases of sepsis; however, when calculated before of the intubation, it is not as useful or predictive of mortality for mechanical ventilated patients, as demonstrated by a 2022 study conducted on patients affected by COVID-19, titled "Preintubation Sequential Organ Failure Assessment Score for Predicting COVID-19 Mortality: External Validation Using Electronic Health Record From 86 U.S. Healthcare Systems to Appraise Current Ventilator Triage Algorithms" conducted by Michael B. Keller et al [27].

- Simplified Acute Physiology Score (SAPS-II): it is a score that measures condition's severity for patients aged over 18 admitted in ICU. It takes into consideration several variables such as age, GCS, chronic diseases and more. The total of parameters taken into consideration are 17 [26], 12 of these are measured within the first 24 hours of the ICU admission. However, once the score is calculated for one patient, it will not be recalculated until a new admission.

- Age, Sex, Comorbidity and others: all these variables have an important influence when it comes to decide whether or not weaning a patient from mechanical ventilation. However, they are either static or binary variables, for this reason, it is not possible to include them in the datasets that has to be used for a model working with time-series: instead the two set of features (time-dependent and static/binary) can be concatenated after working on them separately [29]. In conclusion, considering that the focus of this work is to correctly impute missing values in time-series data, these variables are left out.

3.2 Creation of final datasets

Once the subsets were created, the following step was to develop a subset containing important features, not only relevant to the task but also significant for the type of imputation used, based on time series. The features must be relevant for the weaning task and for time-series imputation. All these analysis were made using MATLAB R2023b, MathWorks, Inc. software.
The first step was to consider only the patients that all the selected subset have in common, they are 8114[2]. This choice was made due to the fact that not having an entire set of variables included in one of the subsets would result in an excessive number of features to replace. Anyways, the fact that the patient is present in one of the subset does not imply that it has a value for every feature, nor that it have a value for the majority of them; actually, it could also be a missing value, considering that they are present in the subsets.

3.2.1 General Feature Selection Criteria

After the number of patients was reduced, all the analysis on the features started. First, every subset was checked to find variables that eventually have only missing values (they are represented by Not a Number (NaN)). Only two variables in Blood Test subset matched this case: *Troponin (ng/L)* which has a total of 16,741 records and all of them are NaNs (for 4,795 patients) and *B-type Natriuretic Peptide (BNP) (pg/mL)* which has 1,560 records and all of them are NaNs (for 1,170 patients). Because of that, these variables have been deleted.

[2]Later on the Drugs subset will be excluded, so the common patients are already presented as the ones in common among the other subsets.

After that, the outliers of every variable have been removed. This was performed by using the formula in Eq. 3.1.

$$\text{Outlier if } |X_i - \text{median}(X)| > 1.5 \times \text{IQR} \tag{3.1}$$

In Eq 3.1 X_i is the value of the element being considered; median(X) is the median of the variable being considered; IQR is the interquartile range defined as third quartile minus first quartile. After the deletion of the outliers for every variable, the following step was finding those variables that have zero variance among all patients, in fact it is always useful to have variety among a variable, but in time-series tasks it is mandatory in order to have informative variables. One variable with zero variance was found in Blood Gas subset:*SpO2 Desaturation Limit (%)* which has 160,308 records and zero NaNs (for 8,122 patients). Another one was in Ventilation: *Positive End Expiratory Pressure (PEEP) set (cmH2O)* which has 323,823 records and none of them is NaN (for 8,096 patients).
Once the not suitable variables were excluded, other criteria were applied in order to obtain a dataset with an admissible number of NaNs. Thus, the next exclusion rule was to delete all the variables that do not have a record for more than 20% of the total patients. Only one variable from the Drugs dataset met this rule, therefore, that entire dataset was deleted. The variables selected with this rule are 15 in Blood Test, 11 in Blood Gas Test, 8 in Vital Signs and 9 in Ventilation, for a total of 43 variables.

3.2.2 Specific Feature Selection Criteria

In order to prepare the data for the necessary time sampling, it is important to determine how many records are appropriate to choose. The first step was calculating the number of records for each variable and for each patient, resulting in a matrix 8114x43. Afterwards, mode, median, mean and standard deviation of the records was computed. The results are presented in Table 3.6.

Table 3.6: Table with statistic parameters computed on the number of records for each variable and each patient.

Standard Deviation	**Mean**	**Median**	**Mode**
92.72	42.06	16	8

At this point, the goal was to obtain an initial dataset with a reasonable number of NaNs, considering they will probably be added later during the time sampling. For this reason, all the variables with a mode, computed on the records of all patients, equal to zero and a median below eight were deleted. These variables are: *Alanine Aminotransferase (ALT/GPT) (U/L)*, *Aspartate Aminotransferase (AST/GOT) (U/L)* and *Bilirubin (mg/dL)* from the Blood Test subset; *Free Calcium (mEq/L)* from the Blood Gas Test subset and *PEEP total (cmH20)* from the Ventilation Subset. At this point the remaining variables

are 38.
Besides the selected variable, other variables can be important in the valuation of the optimal time for weaning, some of which were presented in Section 3.1.2.6. The majority of them have been excluded because they are static or binary (age, sex etc); while for others variables like SAPS II and SOFA only one record for patient has been provided, which makes impossible including these variables in a time-series dataset, because they should be treated as static variables. GCS on the other hand, has more records for each patient, so it has been added to the 38 variables previously selected.
At this point, to reduce even more the number of NaNs, patients with more than 19 variables missing were deleted. The total number of patients went from 8114 to 8101. The choice of keeping as many patient as possible is due to the classification task, which may be penalized if the number of examples for class are unbalanced. The 8101 patients are now subdivided in: 6059 (74.71%) for class one (Extubation Success) and 2049 (25.29%) for class zero (Extubation Failure). A schematic view of the procedures followed to select variables and patients is in Fig. 3.2.

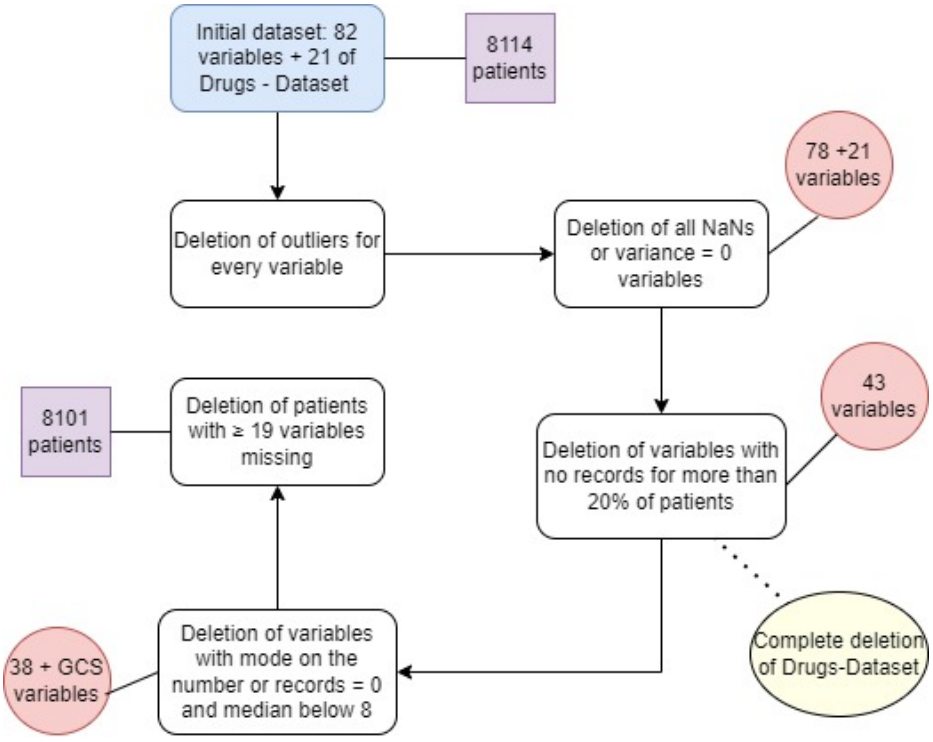


Figure 3.2: A schematic overview of the process followed to select patients and variables for the creation of the new dataset.

3.2.3 Time Sampling

In order to obtain the distances between records (Deltas), the dates were sorted. For each variable associated with the i-th patient, the earliest date was

identified as the first recorded instance of that variable. This selected record served as a reference point, and the time of each subsequent record for that variable was calculated by subtracting the reference time from it. This process was applied to every variable for the i-th patient and subsequently for all 8101 patients. An alternative method for computing the deltas values would be to take the first record among all variables for the i-th patient and set it as a reference value; then, to all the other variables would be subtracted this reference value. However, multiple studies [15], [31] have utilized the first method to preserve the time dependence among the same variable, which may be less strong using the second method. According to this studies, the first method was chosen, and all the results were saved in Comma Separated Values (*csv*) files, one for each patient. The file obtained contains: SubjectID, ItemID, Parameter Name, Value, and Deltas calculated previously.
In a study titled "Prediction of prolonged mechanical ventilation in patients in the intensive care unit A cohort study" conducted in 2013 [30] the 20% of recruited patients were mechanical ventilated for less than three days while 50% stayed on mechanical ventilation less than 7 days. There is no specific duration of mechanical ventilation that is more common than others; in literature, ventilation time is related to the particular study. For this reason, as a first step the ventilation time of the patients was analyzed and the result is shown in Fig 3.3.

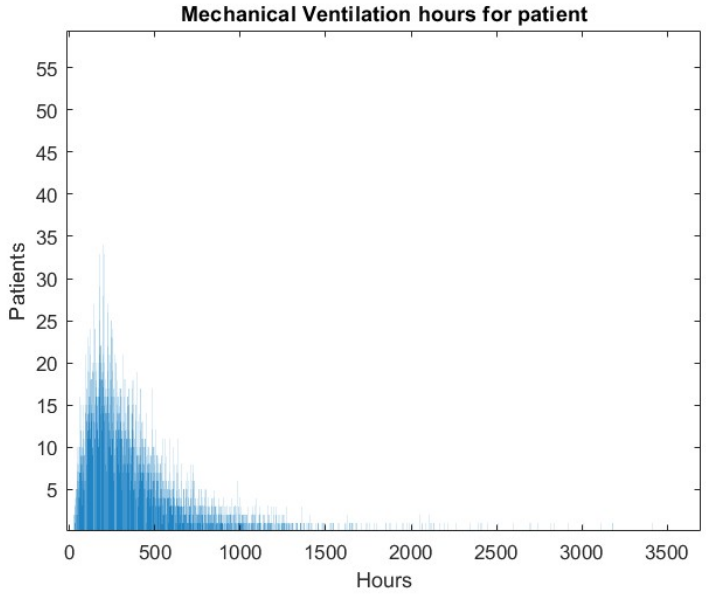
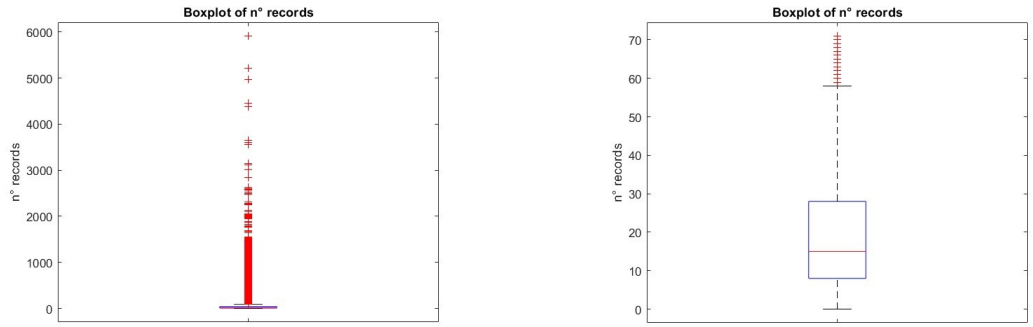


Figure 3.3: The figure shows the time of mechanical ventilation for all the patients included in the dataset.

Fig.3.3 shows that the duration of mechanical ventilation for the patients included in this dataset is sparse, but it is possible to conclude that the majority of patients remain on ventilation for less than 500 hours. However, an analysis of mechanical ventilation duration showed that only 401 patients have records

for more than 250 hours but less than 500. Including 500 hours of records for patients with less than 250 hours would introduce a significant amount of missing values to the subset. On the other hand, the aim of this thesis is not to accurately classify patients in terms of extubation success or failure, but rather to impute missing values in order to create a dataset that can be used to successfully predict that classification. Considering this, all patients have been included regardless of their hours of intubation. However, due to this choice, the patient classes are not expected to aid the imputation process as much as initially anticipated. This is because for example, for a patient who is weaned after 500 hours it is not crucial the evaluation of the first 48 hours since he was intubated, instead it is important to analyze the records closer to his extubation time.



(a) Boxplot with outliers

(b) Boxplot without outliers

Figure 3.4: The images represent the distribution of the number of records calculated on a set containing the records for each variable and each patient. In Fig. 3.4a there is a representation containing all the values, which is extremely sparse; while in Fig. 3.4b all the outliers have been removed using the same formula presented in Eq. 3.1.

Because of the above mentioned reasons and the number of records for each patient (Fig 3.4) , it seems reasonable to select an initial sampling period of 48 hours, starting from the patient's first record for each variable. This choice is justified because opting for a sampling period of 7 or more days would require far more samples than those available for each patient. This would inevitably lead to the substitution of the missing records with NaNs values, surely increasing the total number of NaNs in the dataset. The obtained *csv* files have each one a different number of rows corresponding to the total records registered for the considered patient. However, for the purpose of this work, every patient must have the same number of records for the same number of variables. Because of this, another process on the files was needed.

The subsequent process on the files consisted in choosing a time interval for the time sampling, the first choice was a 0 to 48-hours time sampling in 2h intervals. To minimize the number of NaNs, a tolerance of ± 1 hour was introduced. This

allowed the value corresponding to the considered time to be accepted not only if it fell within the 2-hour interval but also if it was within the range of 1 hour before or after that interval. Additionally, all the variables had to be present in each patient's file, for this reason, it was also checked; if a value was not present for the considered hour, a NaN was added to fill the gap. In the end, each patient has a file of 25 (sequence length from 0 to 48 at 2h intervals) for 39 variables, resulting in a total of 975 rows plus one for naming the columns. The structure of the new *csv* files is the same as the previous one, but this time each patient has the same number of rows.

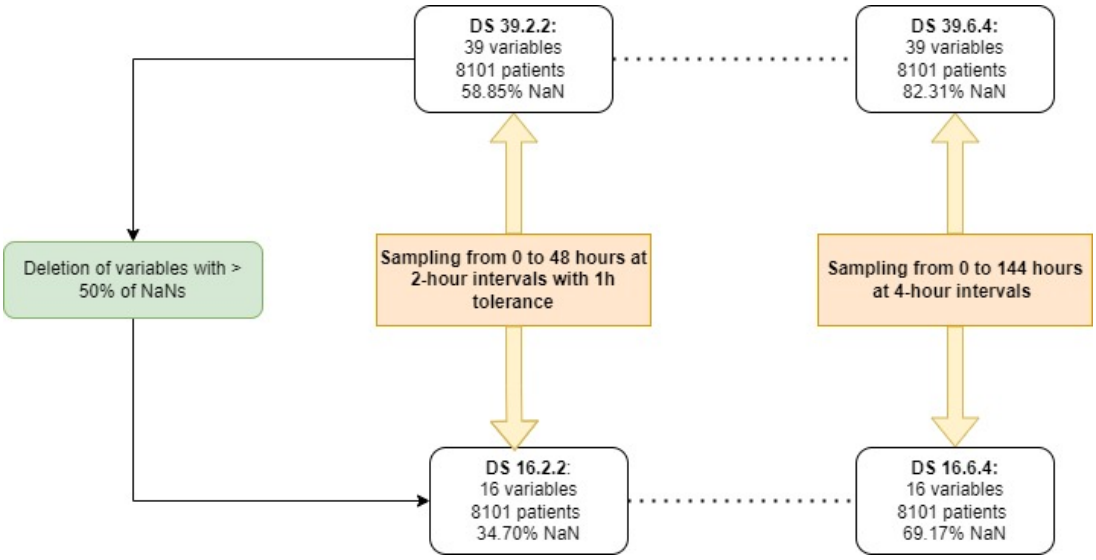


Figure 3.5: Overview of the process followed to obtain the four datasets.

The percentage of NaNs on the obtained dataset (DS 39.2.2) is 58.85%. In order to further reduce the number of NaN values, an additional check was performed on the obtained dataset. Every variable with a percentage of NaN values greater than 50% of its total records, among all patients, was removed, obtaining a dataset of 16 variables (DS 16.2.2) (Table 3.7). The dataset obtained was sampled like the previous one, in this case the total number of rows for each patient is 400 + 1; the percentage of missing values was reduced to 34.70%.
In order to include more data and hence having a more real-world based dataset, with different percentages of missing values to compare, the data of more than three days should be included. In fact, according to the paper previously mentioned [31], around 50% of the mechanical ventilated patient stayed intubated for more than three days and less than seven, so it seems a plausible range of time to analyze. Furthermore, in the provided dataset 276 patients have stayed mechanical ventilated for less than three days, while 1276 stayed ventilated for more than three days and less than seven. Meaning that, in theory, the labels which indicate the outcome of weaning would be more helpful for those patients. These consideration led to the creation of two additional datasets, sampled from 0 to 144 hours with a 4 hours time interval. In this case, no tolerance was added

in order to better understand the performances of the methods that are going to be applied. The two datasets obtained differ in the number of variables, DS 39.6.4 has 39 variables, so each patient has 1443 + 1 rows and a percentage of missing values of 82.31%; the second dataset DS 16.6.4 has 16 variables, a total number of rows for each patient of 592 + 1 and a percentage of missing values of 69.17%.

Table 3.7: This table contains the variables selected for the final dataset. Those contained in a red box are the ones selected for the datasets with a reduced number of NaNs. All the variables are taken from the original subsets presented in Chapter 3, except from GCS.

Variable with Unit of Measurement	**Variable with Unit of Measurement**
Chloride – Blood - Chemistry (mEq/L)	Creatinine (mg/dL)
Glucose – Blood - Chemistry (mg/dL)	Potassium - Blood - Chemistry (mEq/L)
Sodium – Blood - Chemistry (mEq/L)	Azotemia (mg/dL)
Hemoglobin - Blood - Hematology (g/dL)	International normalized ratio (INR)
Platelets (k/uL)	Prothrombin time (s)
Partial thromboplastin time (s)	White blood cells (k/uL)
Base Excess (mmol/L)	Lactic acid (mmol/L)
pCO2 (mmHg)	pH
pO2 (mmHg)	Bicarbonate (mEq/L)
Total Calcium (mEq/L)	Arterial O2 pressure (mmHg)
Arterial CO2 pressure (mmHg)	Tidal Volume spontaneous (ml)
Plateau pressure (cmH2O)	
O2 saturation pulseoxymetry (%)	Heart Rate (bpm)
Non invasive blood pressure systolic (mmHg)	Non invasive blood pressure mean/diastolic (mmHg)
Respiratory Rate (insp/min)	Temperature (°C)
Respiratory Rate set (insp/min)	Respiratory Rate spontaneous (insp/min)
Respiratory Rate total (insp/min)	Fraction Inspired Oxigen (FiO2) (%)
Tidal Volume set (ml)	Tidal Volume observed (ml)
Minute Volume (L/min)	Positive Inspiratory Pressure (PIP) (cmH2O)
Mean Airway Pressure (cmH2O)	Glasgow Coma Scale (GCS)

3.3 BRITS

The Bidirectional Recurrent Imputation for Time Series is the choice among the other methods presented in Chapter 2. In fact, a review from 2023 titled "Deep imputation of missing values in time series health data: A review with benchmarking" by Maksims Kazijevs and Manar D. Samad [28] compared the performances on healthcare datasets of multiple state of the art methods to impute missing values. The study revealed that BRITS and CATSI are promising networks compared to the other models considered. Other methods analyzed include TAME and NAOMI, which achieved optimal results but required ground-truth data for training. For this reason, they were not considered in this work. CATSI, on the other hand, does not require ground-truth for training and demonstrated performance comparable to BRITS. However, in the reference study BRITS showed better performances when the missing rate was above 50% with every type of missing value (MNAR,MAR,MCAR). CATSI, on the other hand, performs better at lower missing rates. Moreover, BRITS seems to be the best method to impute MNAR, in comparison to CATSI, with every percentage of missing values. For these reasons, BRITS is more suited to the purpose of this work and is therefore the network of choice. The implementation was carried out using PyTorch and is based on the source code provided by the authors of BRITS on GitHub[3].

3.3.1 Creation of *json*

First step was to organize all the data. The network required this data as a list of dictionaries with multiple keys in a file *json*. The script functions as intended: it loads the pre-prepared *csv* files and extracts the hours from the deltas column, while ignoring the minutes and seconds. The script then collects data corresponding to the specified sampling times. The values are normalized using standard deviation and mean of the reference variable (Eq. 3.2).

$$\text{Normalized Value} = \frac{\text{Original Value} - \text{Mean}}{\text{Standard Deviation}} \tag{3.2}$$

Normalization occurs because variables are in different ranges and units, this passage allows the model to treat them uniformly during training.
The metrics calculated to evaluate the performance of the model, discussed later in this paper, need a ground-truth. Because of that, the model randomly hides 10% of the non missing data and marks them as missing, these values are then used to evaluate performances.
Each patient's data is stored as a dictionary with three keys:

- ***label***: this contains the patient outcome (1 for successful weaning, 0 otherwise). This information is provided by a *.txt* file, created with MATLAB, which contains every patient Subject ID with its corresponding label.

[3]https://github.com/caow13/BRITS

- ***forward*** and ***backward***: each contain a dictionary with other information about the patient. In the first case data is processed forward in time; while in the second, data is processed in reverse. Both the dictionaries contain the same kind of information:
  - **values**: contains the values observed and missing as NaN that are placeholders for future imputation. They are either in their original order or reversed (it depends on which dictionary is taken into consideration).
  - **masks**: contains binary masks showing if a value is available (1) or missing (0). The mask is based on the values contained in "values".
  - **evals**: contains the true values for the imputation evaluation. So it contains the real data without the "fake" missing values that have been added in "values".
  - **eval masks**: contains the masks for evals, this indicates which values are actual NaNs and which are added for evaluation purposes.
  - **deltas**: provides a way to quantify changes over time for each variable, its final structure will be a 2D array, where each row corresponds to a time step and each column correspond to an attribute. In detail, to calculate deltas, the first step is initialization, at the first time step, delta is initialized as a vector of ones, of the same dimension as the number of variables, hence assuming each variable have been observed in the previous step. For each following hour, the delta is calculated based on whether the record was observed or not; this check is done utilizing the masks mentioned above. If a value is observed, the delta for that variable is reset to one, meaning the time from the last observation is zero. Otherwise, if the value is missing, the delta is increased by adding one to the previous delta. This implementation give the model information about how much time has passed since the last observation for each attribute. This helps the model handle missing data more effectively. Fig 3.6 shows an example of how the deltas vector is calculated.
  - **forwards**: the original version of BRITS includes this because they used a GRU-D model, as a method of comparison to their own. GRU-D required this dictionary, but it is not used in BRITS. It contains a different version of "values" in which the NaNs are replaced using the last value observed.

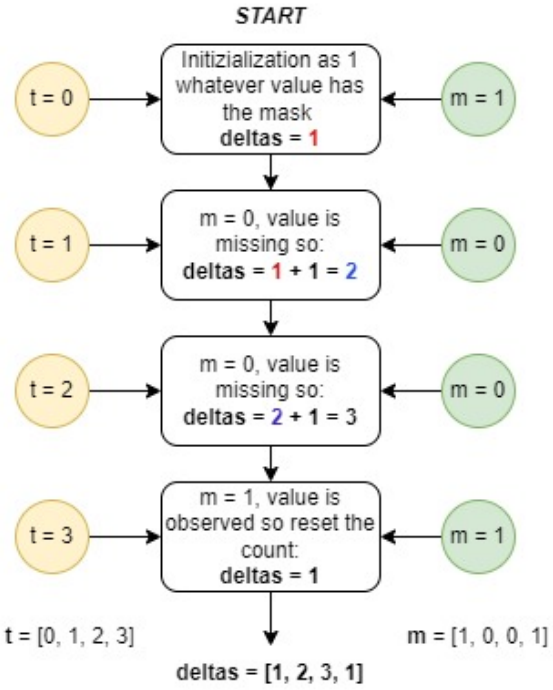


Figure 3.6: Example of calculation of deltas vector for four values. The mask is indicated with "m", while the vector of times is indicated with "t".

3.3.2 Algorithm

A standard neural network is represented by Eq, 3.3:

$$h_t = \sigma(\mathbf{W_h} h_{t-1} + \mathbf{U_h} x_t + b_h) \tag{3.3}$$

In Eq. 3.14 h_t is the hidden state at time t. σ is the activation function (sigmoid). $\mathbf{W_h}$, $\mathbf{U_h}$ and b_h are parameters, respectively the weight matrix that multiply the previous hidden state, the weight matrix that multiply the input vectors and the bias vector. x_t is the input vector at time t. The first hidden state is initialized as an all zero vector. After that, through the regression component the estimated vector is calculated as in Eq. 3.4.

$$\hat{x_t} = \mathbf{W_x} h_{t-1} + b_x \tag{3.4}$$

Considering that in BRITS x_t has missing values and, as it has been said, one of the strengths of this method is the fact that it does not need to initialize them, some method to replace them is needed because it is impossible to use the vector raw. For this reason a "complement" vector (Eq, 3.5) is used whenever there are missing values.

$$x_t^c = m_t \odot x_t + (1 - m_t) \odot \hat{x_t} \tag{3.5}$$

In this vector the missing data in x_t, that are found with the use of the mask m_t, are replaced by the corresponding values in $\hat{x_t}$. However, this implementation assumes that the variables are mutually uncorrelated, in order to fix this it is

introduced a new term. From now on, $\hat{x_t}$ is the history-based estimation, now a feature-based estimation will be added (Eq. 3.6).

$$\hat{z_t} = \mathbf{W_z} x_t^c + b_z \tag{3.6}$$

W_z and b_z are the corresponding parameters. In particular W_z has the diagonal of all zeros. In this way, the i-th element, considering I the set containing all the variables, in $\hat{z_t}$ is the estimation of the the x_t^i based on the other features. The two estimations are then combined, to do so other terms are needed to add to the equation, like the *temporal decay factor* (Eq. 3.7).

$$\gamma_t = exp(-max(0, W_\gamma \delta_t + b_\gamma)) \tag{3.7}$$

In Eq. 3.18 γ_t is the temporal decay factor while δ_t is the deltas at time t. Once the temporal decay factor is calculated the next step is to calculate the weight β_t (in a range between 0 and 1, (Eq. 3.8)) that combines the history based estimation $\hat{z_t}$ with the feature-based one $\hat{z_t}$. The feature based vector is derived from the complement vector as it is previously shown. However, it is not mandatory for the complement vector to contain history-based estimation because where there is the true value, there is no need for an estimation.

$$\beta_t = \sigma(W_\beta(\gamma_t \bullet m_t) + b_\beta) \tag{3.8}$$

Then we can calculate the combined vector as in Eq. 3.9:

$$\hat{c_t} = \beta_t \odot \hat{z_t} + (1 - \beta_t) \odot \hat{x_t} \tag{3.9}$$

At this point we can obtain the combined complement vector (Eq. 3.10).

$$c_t^c = m_t \odot x_t + (1 - m_t) \odot \hat{c_t} \tag{3.10}$$

$$h_t = \sigma(W_h(h_{t-1} \odot \gamma_t) + U_h(c_t^c \bullet m_t) + b_h) \tag{3.11}$$

This is the process followed to obtain the hidden vector h_t (Eq. 3.11). This sequence is followed for forward and backward directions, in the end a sequence of estimations is obtained and, in order to obtain the estimation at time $t-$th, the mean of the two estimations is calculated.

3.3.3 Loss Calculation

The loss function represents a mathematical method of evaluating how well the algorithm is modeling the dataset. The lower the better the model, so the goal is to minimize it as much as possible. It guides the learning algorithm in adjusting the weights during the training. The total loss at time $t-$th for the imputation task is calculated as in Eq. 3.12.

$$l_t = L_e(x_t, \hat{x_t}) + L_e(x_t, \hat{z_t}) + L_e(x_t, \hat{c_t}) \tag{3.12}$$

L_e is the Mean Absolute Error (MAE) between the two values at $t-$th, time step, used to avoid negative values. The corresponding formula for time step t and for the first term of the equation is shown in Eq 3.13; however, it stays the same for the other terms. Assuming N the number of samples in the batch and D the number of features for each sample.

$$\mathrm{x}_{loss}^{t} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{D}(|x_{i,j,t} - \hat{x}_{i,j,t}| \cdot m_{i,j,t})}{\sum_{i=1}^{N}\sum_{j=1}^{D}(m_{i,j,t}) + 10^{-5}} \tag{3.13}$$

In Eq. 3.13, m represents the mask and a small number is added to avoid division by zero which may cause errors in the code.
The decision of using the loss in Eq. 3.12 instead of taking only the third term of the equation $L_e(x_t, \hat{c_t})$, which represent the final imputation compared to the real value, is due the fact that the convergence was too slow using only this term. The loss is calculated this way for both the forward and backward direction.
Another term is then added to this equation, the one about the classification task, in this case the loss is calculated as the binary cross-entropy, which is a common method used as loss function in classification tasks. Usually the binary cross-entropy is calculated as in Eq. 3.14:

$$\text{loss} = -\left(y \cdot \log(\sigma(x)) + (1-y) \cdot \log(1-\sigma(x))\right) \tag{3.14}$$

In Eq. 3.14 y is the target label (0 or 1); x is the logit (meaning the raw model output); $\sigma(x)$ is the sigmoid function, which is necessary to transform the raw value into a probability. However, in this case it has been used a loss for binary classification that directly apply to logits without requiring a separate sigmoid activation (Eq. 3.15).

$$\text{loss} = x - x \cdot y + \max(0,-x) + \log\left(\exp(-\max(0,-x)) + \exp(-x - \max(0,-x))\right) \tag{3.15}$$

In Eq. 3.15 max(0, -x) and the log terms improve numerical stability when working with large or small values of x; for this reason it is preferred to use this formula in deep learning tasks.

At this point the term corresponding to the imputation and that corresponding to the classification are united to make possible having one unique term for the loss function in backward direction and one in forward direction (Eq. 3.16).

$$\text{loss}_t^f = \text{Imputation Loss} \cdot \text{Impute Weight} + \text{Classification Loss} \cdot \text{Label Weight} \tag{3.16}$$

Impute Weight and *Label Weight* are the parameters that can be changed to establish which term should have a bigger influence on the total loss, while *Imputation Loss* and *Classification Loss* are Eq. 3.12 and Eq. 3.15 respectively.

In conclusion, one last term is added to calculate the final loss, this term permits to take into consideration the discrepancy between the forward and backward prediction (Eq. 3.17).

$$l_t^{cons} = \text{Discrepancy}(\hat{x}_t, \hat{x}'_t) \tag{3.17}$$

$\hat{x_t}$ is the forward estimation at time $t-$th, while $\hat{x}'_t$ is the backward estimation at the same time step. Once the last term is added, the final loss is represented by the sum of the three equations: Eq. 3.16 for the backward direction, the same equation for the forward direction and Eq. 3.17.

$$\text{Final Loss}_t = \text{loss}_t^f + \text{loss}_t^b + l_t^{cons} \tag{3.18}$$

3.3.3.1 Deletion of Classification Task

While combining classification and imputation is one of the strengths of BRITS, the main purpose of this work is to correctly impute missing values. Considering the unbalanced classes, the fact that not all the useful variables to perform classification task are included in the datasets utilized and that the time sampling not corresponds to the crucial hours that precedes extubation for each patient, the classification task may result harder, maybe even not helpful to the imputation process. For this reason, it has been decided to try the model with and without the labels. This obviously has an impact on the loss, which means the imputation process has been impacted. The Eq. 3.12 stay the same, but Eq. 3.16 changes to Eq. 3.19.

$$\text{loss}_t^f = \text{Imputation Loss} \cdot \text{Impute Weight} \tag{3.19}$$

Then the final loss is calculated as in Eq. 3.18, but the forward and backward losses are different.

3.3.4 Parameters and Structure

In Fig. 3.7 there is a scheme of functioning of the model with all the layers implemented and their dimensions; while in the rest of this subsection some parameters will be presented and explained.

- *batch size*: it represents the number of data samples processed before the model updates its weights during training. A small batch size can lead to noisy predictions but updates the weights more frequently, this can help the model to better generalize. A large batch size offers more accurate estimates but takes more memory and updates less frequently, this can lead to slow convergence.

- *Epochs:* when an epoch ends, it means the model has seen every sample in training set once. If the number of epochs is too low, the model may underfit, meaning it hasn't trained long enough to capture the underlying patterns in the data. On the other hand, if the number of epochs is too high, the model may overfit, meaning it becomes too tailored to the training data and loses its ability to generalize well to new, unseen data. Additionally, training for too many epochs can be inefficient, as it consumes unnecessary computational resources without improving performance on the validation or test sets.

- *Hidden size*: it represents the number of neurons in the hidden layers of a neural network, in LSTM, is the size of the hidden state vector. It is a critical parameter because if the number is too small the model can underfit, on the other hand, too many units can lead to overfitting.

- *Label Weight:* this parameter is specific for this network, it is used to adjust the weight given to the loss associated with predicting the label. It is important to correct balance this parameter to give it the right weight compared to the imputation part.

- *Imputation Weight:* this parameter is also specific for this network. It balances the imputation loss within the total loss during model training, ensuring that the imputation task contributes appropriately to the overall optimization process.

- *shuffle:* it can be set to "True" or "False". It determines whether or not to shuffle the data at the beginning of every epoch; if it is set to true the data is randomized before every epoch. Generally, it is better to shuffle, because it decreases the risk of overfitting, in fact, the model cannot learn order-specific features, however, shuffling introduces randomness that can slower the convergence process.

- *Dropout*: it is a regularization technique to prevent overfitting by randomly setting some weights to zero during training. This means that the weight set to zero are not updated, preventing neurons to rely on each other too much and improving generalization, ensuring a robust model.

- *Optimizer:* the optimization algorithm is Adam (Adaptive Moment Estimation), it is used to update weights of the neural network during training. Its main purpose is to minimize the loss function by adjusting the parameters of the model in response to the computed gradients.

- *Learning Rate:* it is a hyper-parameter that controls how much to change the model parameters in response to the estimated error each time the model weights are updated. An higher learning rate may speed up training but can lead to divergence. In contrast, a lower learning rate may provide more precise convergence to a minimum but can possibly get stuck in a local minima.

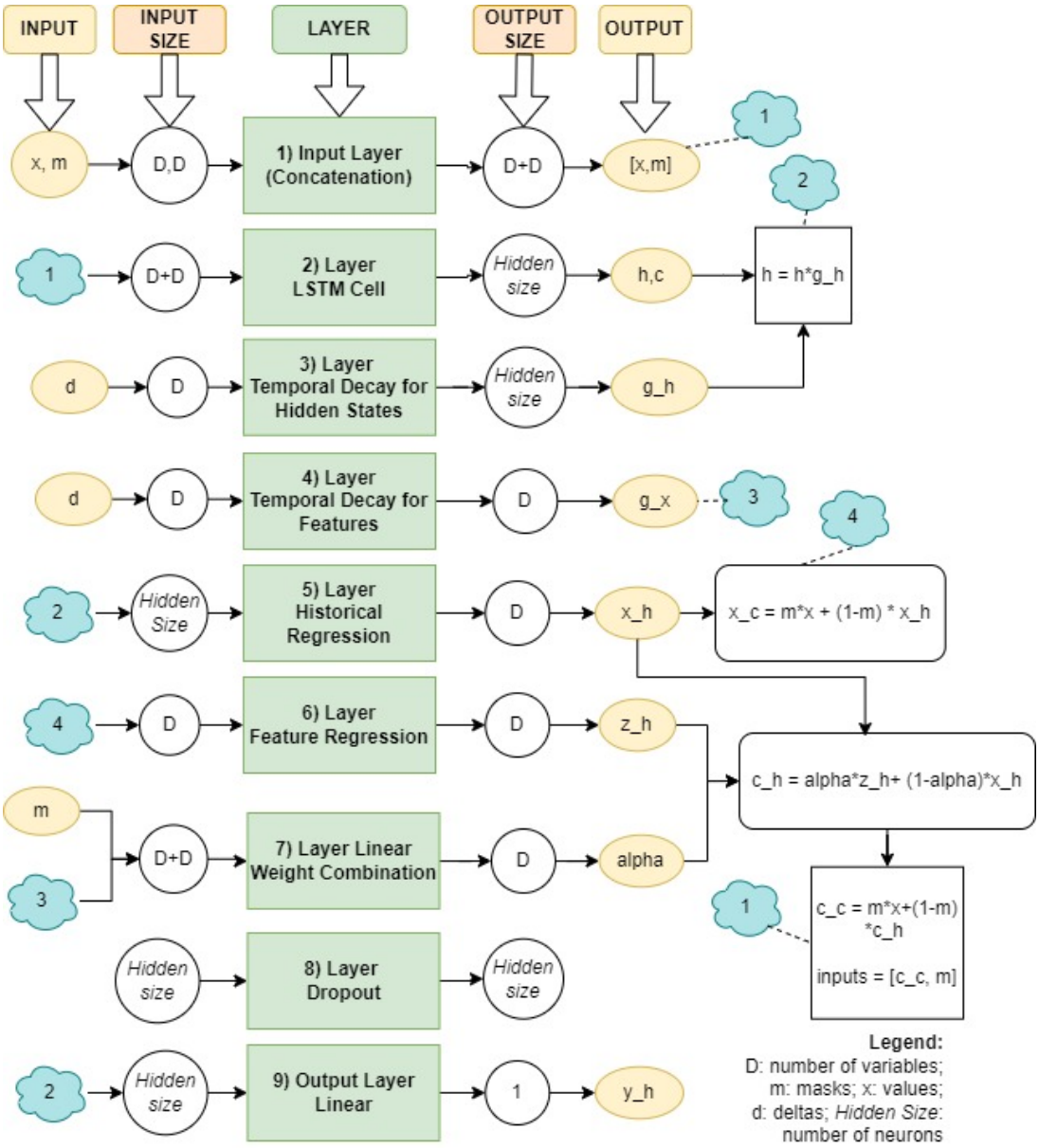


Figure 3.7: Scheme of the algorithm for a single time step, this algorithm is repeated in loop until the sequence length is completed. The number on the layers is to enumerate them not to impose the order. The blue clouds are connected using dotted line to the output they represent because subsequently those become the input for other layers.

3.4 M-RNN

Multi-directional Recurrent Neural Network imputes the missing values using LSTM as its core. This model is developed by the same authors of BRITS [15] . It also has a prediction and imputation part that are structured slightly different from BRITS. In fact, while it is also bi-directional (forward and backward), it treats the imputed values as constants, not considering the possibility of correlation among variables.
To correctly function this method uses the same *json* file as BRITS with exactly the same structure. This model was also implemented using PyTorch and it is based on the same source code of BRITS.

3.4.1 Algorithm

The input arguments, taken from the *json* file, are: x_t which is the input vector for the total number of features D; m_t is a binary mask, used to know where the values is observed or missing; d_t is the time delta. All three are considered at time step t. These values are concatenated to obtain the vector i_t in 3.20.

$$\mathrm{i}_t = concat(x_t, m_t, d_t) \tag{3.20}$$

Afterwards, the LSTM cell updates its hidden state and cell state for each time step t based on the input vector in 3.20 and the previous hidden state (h_{t-1}) and cell state (c_{t-1}) as it is shown in 3.21.

$$\mathrm{h}_t, \mathrm{c}_t = LSTMCell(i_t, (h_{t-1}, c_{t-1})) \tag{3.21}$$

The Eq.3.21 is calculated both in forward and backward direction. Then, at each time step the two resulting equations are concatenated to form a combined hidden state as in 3.22.

$$\mathrm{h}_t = concat(h_t^f, h_t^b) \tag{3.22}$$

The dimension of h_t is double the hidden size imposed as parameter (meaning double the neurons imposed).
Then, there is the Feature and History Regression. At each time step t feature and history regression are computed as shown in Eq.3.23 and Eq.3.24 respectively.

$$\mathrm{x}_f = W_{feat} x_t + b_{feat} \tag{3.23}$$

$$\mathrm{h}_t = W_{hist} h_t + b_{hist} \tag{3.24}$$

In Eq. 3.23 and Eq. 3.24, the W is a matrix of weights used for feature-based estimation and history-based estimation respectively; while b is the vector of biases. Once the two different estimations are computed, they are combined using a learned weight combination (α) for each time step (α_t) (Eq.3.25). The weights (α) are in a range that goes from 0 to 1 and are computed using sigmoid function σ.

$$\alpha_t = \sigma(W_\alpha \cdot concat(m_t, d_t) + b_\alpha) \tag{3.25}$$

$$\mathrm{x}_c = \alpha_t \cdot x_h + (1 - \alpha_t) \cdot x_f \tag{3.26}$$

In Eq. 3.26 x_c is the imputed value at time step t. This equation concludes the imputation part. For the classification part, the entire sequence of imputed values is passed through an LSTM for the final prediction. This output at the final time step is then passed through a linear layer and a sigmoid activation to produce the predicted probability ($\hat{y}$) for binary classification (Eq. 3.27).

$$\hat{y} = \sigma(W_{out} h_T + b_{out}) \tag{3.27}$$

In Eq. 3.27 W_{out}, b_{out} are the weights and biases, respectively, of the final linear layer.

3.4.1.1 Differences with BRITS

In the M-RNN model the imputation is performed independently at each time step and for each variable. This means the model does not take into account how one feature's value could influence another feature's value when imputing missing values.
The feature-based estimation is a linear regression over the current feature value, and operates independently for each variable, not incorporating information from other variables. The same goes for the history-based estimation.
In contrary, BRITS uses backward and forward propagation to propagate the information through time, this helps capture eventual correlation among features. In fact, information from one variable can affect the imputation of another through the shared hidden state. Because of this approach, the model can find correlation between variables analyzing their temporal dynamics.

3.4.2 Loss Calculation

As it works for BRITS, also the M-RNN uses a combination of the imputation loss and the classification one. The imputation loss, which measures how well the model predicts the missing values for each feature over time is computed on the absolute difference between the imputed values and the observed ones. In particular, while in BRITS the imputation loss is a sum of a loss of three terms, in this case we have a single term as shown in Eq.3.28 for time step t.

$$\mathrm{x}^t_{loss} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{D}(\left|x_{i,j,t} - x^c_{i,j,t}\right| \cdot m_{i,j,t})}{\sum_{i=1}^{N}\sum_{j=1}^{D}(m_{i,j,t}) + 10^{-5}} \tag{3.28}$$

In Eq. 3.28 x^c corresponds to the output obtained in Eq. 3.26, while N is the number of samples contained in the batch size and D is the total number of features for each sample. m is the mask to count the observed values and then there is a term added to avoid a division by zero, as in BRITS.
Once the imputation loss is calculated the following step is to calculate the classification loss. It is computed exactly like in BRITS so the steps to obtain it are shown in section 3.3.3.

Once both the imputation and classification loss are obtained they are combined to create the final loss (3.29).

$$\text{loss}_t = \text{Imputation Loss} \cdot \text{Impute Weight} + \text{Classification Loss} \cdot \text{Label Weight} \tag{3.29}$$

This model, like BRITS, has been used both with and without the classification task. When the classification task is excluded, the loss function is modified as in Eq. 3.19.

3.4.3 Parameters and Structure

The parameters are the same showed in Section 3.3.4, but the structure of the network is different, despite using the same core which is LSTM. A scheme of the algorithm is presented if Fig. 3.8.

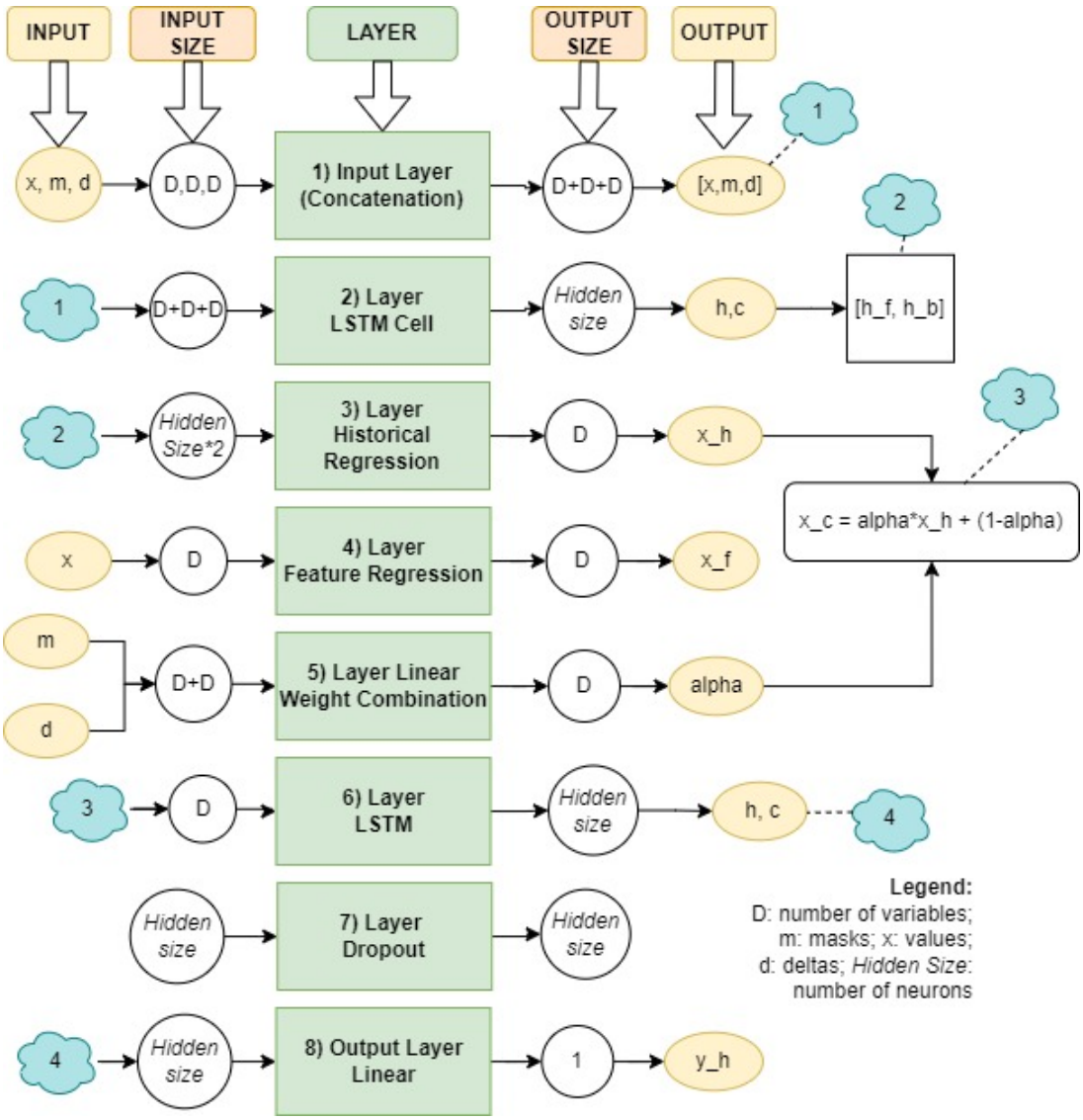


Figure 3.8: A schematic view of the algorithm of M-RNN. The number on the layers is to enumerate them not to impose the order. The succession represents the steps inside the loop of the Sequence Length. The blue clouds are connected using dotted line to the output they represent because subsequently those become the input of other layers.

3.5 Metrics and Loss Function

The metrics used to evaluate the performances on the imputation task of each method are Mean Absolute Error (MAE) which is the average magnitude of errors in a set of predictions, without considering their direction, and Mean Relative Error (MRE) which measures the average relative difference between actual values and predicted values. They are calculated as in Eq. 3.30 and Eq. 3.31.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \tag{3.30}$$

$$\text{MRE} = \frac{\sum_{i=1}^{n}|x_i - \hat{x}_i|}{\sum_{i=1}^{n}|x_i|} \tag{3.31}$$

In Eq. 3.30 and Eq. 3.31, x_i is the observed value while $\hat{x}_i$ is the imputed value. n represents the total number of samples. The lower these metrics the better the imputation.
To evaluate the classification task the metric used is Area Under the ROC Curve (AUC), which represents the area under the Receiver Operating Characteristic (ROC) curve, which plots the True Positive Rate (sensitivity) against the False Positive Rate (1-specificity) at various threshold settings. In order to better understand this metric the True Positive Rate (TPR) and False Positive Rate (FPR) are shown in Eq. 3.32 and Eq. 3.33, while their terms are explained in Fig 3.9.

$$\text{TPR} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{3.32}$$

$$\text{FPR} = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}} \tag{3.33}$$

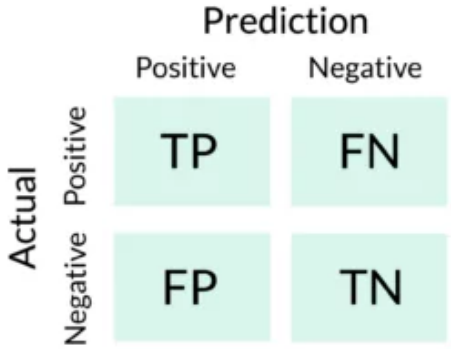


Figure 3.9: This image can be useful to better understand what the terms used in binary classification mean. Picture taken from "What is the True Positive Rate in Machine Learning?" by Iguazio Ltd.

The higher this metric the better the classification. In particular:

- AUC = 1: the classifier is perfect
- AUC = 0.5: the performance of the classifier corresponds to random guess the classes.
- AUC < 0.5: the classifier is not good, in fact its performance is worse than random guessing.

In regard to the loss function, it is useful for understanding how to set the model's parameters and evaluate its performance in terms of weight adjustment. The primary objective is to minimize the loss function as much as possible.
In this thesis, the loss function used to assess the model and set the parameters is based on the average loss. Each model has its own method for calculating the loss, as discussed in Chapter 3. The calculated loss for each model is then processed to derive the average loss function. While this average loss function is not directly useful for the model itself, it serves as an indicator of the model's performance over time. The average loss is calculated by initializing to zero a variable *"run loss"* which will contains the losses. During each batch iteration, the model returns the loss for that batch, which is added to *run loss* accumulating the total loss for all batches in the current epoch. At the end of each batch, the average loss is calculated by dividing the total loss (*run loss*) by the number of batches seen so far.

It is important to understand how the model is performing, not only for the results, but also for its capacity of generalization. In fact, once a model is trained, this should be able of working with unseen data to give more or less the same performances. To achieve that, it is important that the model do not underfit nor overfit. Overfitting a model occurs when a model adapt to the training data, losing its generalization capacity, so the results will be optimal on training but not good enough on unseen data. On the other hand, underfitting happens when the model does not learn enough from the training data resulting in poor performances both on training and unseen data. Generally, to evaluate if a model is overfitting or underfitting, the training and validation loss are compared, because the comparison is useful to understand if there is discrepancy between the performances on training and validation data.
However, in this thesis, the training and validation loss is not calculated. This is because, these concept cannot adapt to the types of models implemented. In fact, since BRITS masks a portion (10%) of the dataset to create challenges in the imputation task, there are not conventional training and validation losses. Thus, the average loss provides a summary of the model's performance across batches that can help in tuning parameters such as batch size learning rate and dropout.

Chapter 4

Results and Conclusions

In this chapter the performances obtained on previously presented methods will be compared and discussed. All the models have been tested on all four the datasets presented: DS 16.2.2, DS 16.6.4, DS 39.2.2, DS 39.6.4.

4.1 BRITS

BRITS is the main model of this thesis and is expected to perform better than M-RNN. All analyses have been conducted with and without the classification component to evaluate different performances and to understand whether using the labels effectively aids in the imputations.
The parameters set for this model are shown in Fig. 4.1.

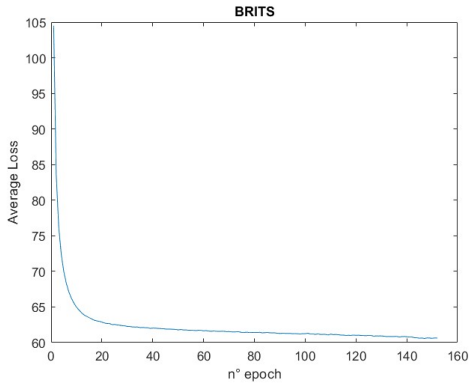
Batch size	64\128
Hidden Size	100\128
N° Epochs	100\200\300
Label Weight	1
Imputation Weight	1\3
Shuffle	True
Dropout	0.25
Learning Rate	10^(-3)

Figure 4.1: Parameters set for this model. Some rows have multiple numbers because the parameters have been set differently for the various cases analyzed. In particular, the imputation weight is different whether the classification part is used (3) or not (1).

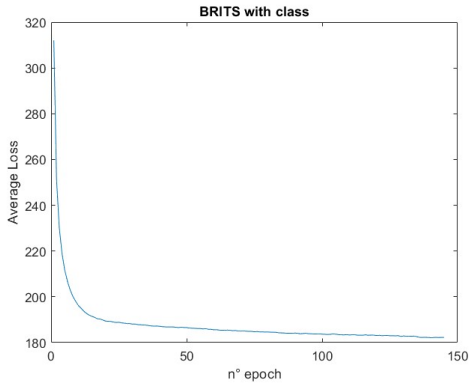
The learning rate showed in Fig. 4.1 is the initial one. In fact, the learning rate changes when there are no improvement in the average loss for more than five epochs (patience = 5); decreasing its value by 90% of the previous one.

In Fig 4.2 is showed a comparison between the model that incorporates classification and the one that does not, in terms of average loss. As shown, the

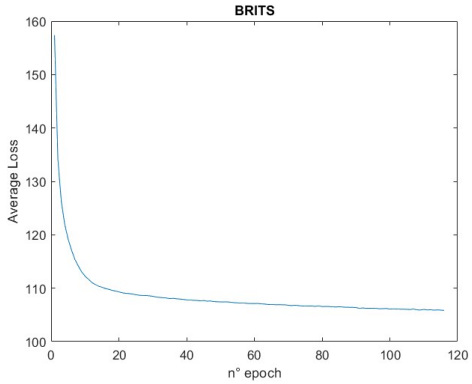
loss values differs numerically, this is because with the classification two weights are required to calculate the final loss: one for the imputation part and one for the classification part. Since the aim of this thesis is primarily focused on the imputation aspect, the weight for this component was tripled compared to the weight for the classification. This adjustment led to an increase in the final loss. For these reasons, a higher average loss does not necessarily indicate that the model is worse. In fact, the comparison should not be based solely on this, but rather on how the loss decreases and stabilizes over time.



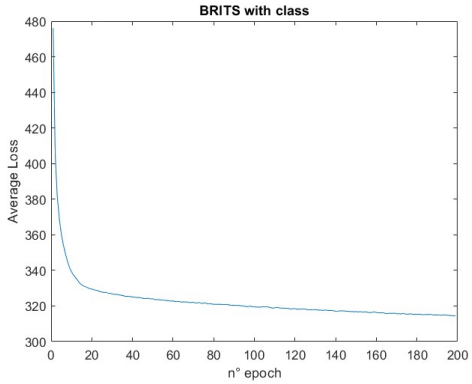
(a) DS 16.2.2 no class



(b) DS 16.2.2 class



(c) DS 16.6.4 no class

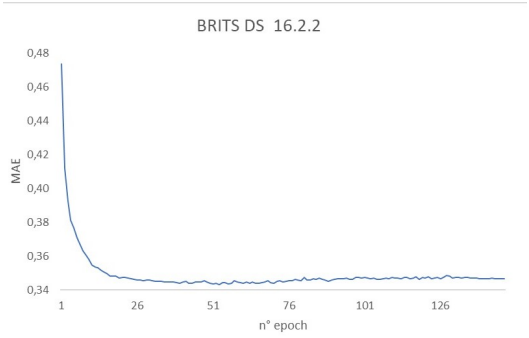


(d) DS 16.6.4 class

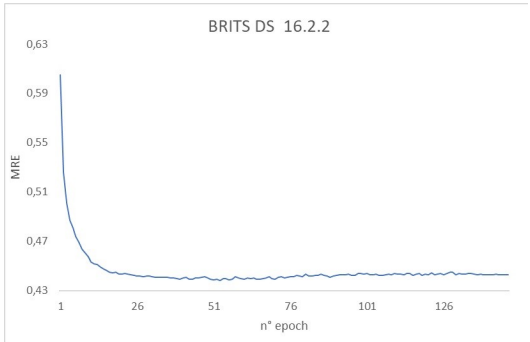
Figure 4.2: The images represent the average loss during the training of BRITS model on dataset with 16 variables. In Fig. 4.2a and in Fig4.2c is showed the average loss when it is deleted from the model the classification part. While in Fig. 4.2b and Fig. 4.2d it is showed the average loss using the classification task, which has an impact on the total loss.

The loss for these models appears stable, as it consistently decreases over time, as expected. There are no clear signs of overfitting or underfitting in the graph. However, the results show a plateau, indicating that the model has reached its optimal performance and no longer improves. This is why training stopped, in some cases, before reaching the predefined number of epochs.

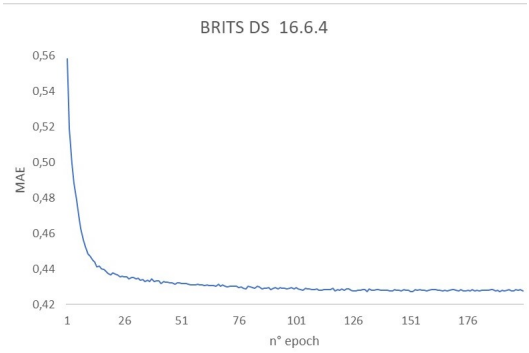
Fig 4.3 presents the results in terms of MAE and MRE for this model. The results do not differ significantly when the same dataset is considered, demonstrating that adding the classification component does not substantially improve the model's performance. However, as discussed earlier in this thesis, although an improvement in performance was anticipated, it was also expected to be minor. While using labels for classification could theoretically enhance the imputation, in this case, the data analyzed is not predominantly from the period closer to extubation. Intuitively, the hours leading up to extubation seem more significant for improving classification accuracy than the hours following intubation. The decision to include as much data as possible was made due to the broader objective of this thesis.



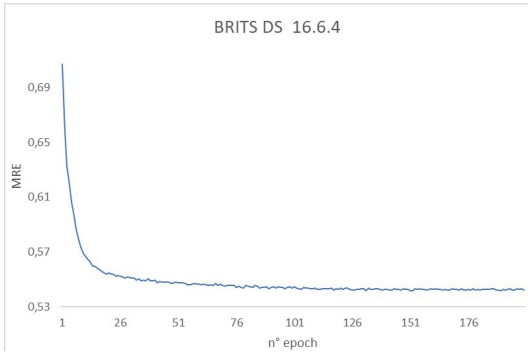
(a) MAE of DS 16.2.2 class



(b) MRE of DS 16.2.2 class



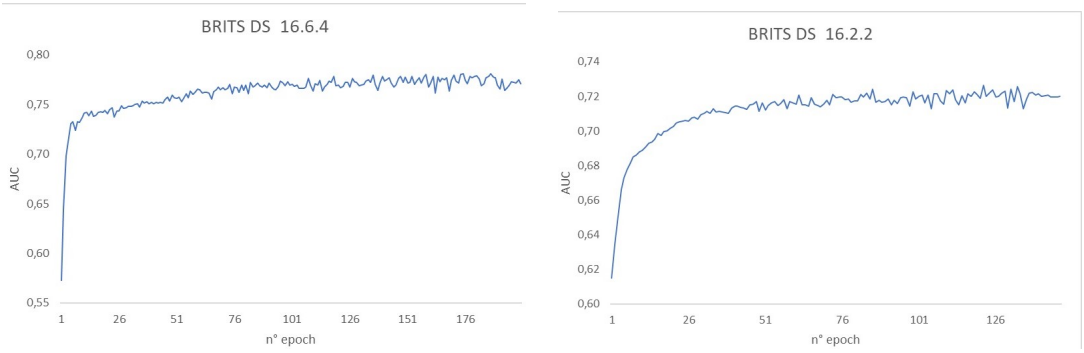
(c) MAE of DS 16.6.4 class



(d) MRE of DS 16.6.4 class

Figure 4.3: The images represent MAE and MRE during the training of BRITS model on dataset with 16 variables. In all the images the model was applied with both imputation and classification parts.

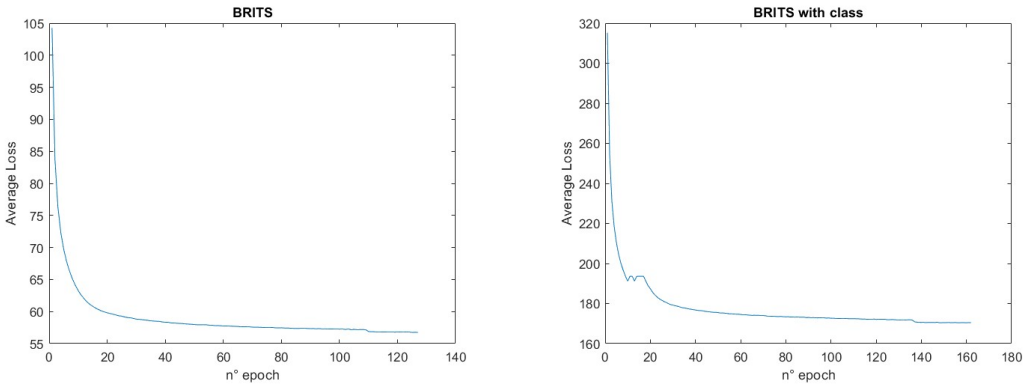
Regarding the classification part, it was penalized in the choice of the weights to assign to each task. This choice reflects on the graphs of the AUC over time (Fig. 4.4). Overall, the results are not bad, as they consistently remain above 50%. However, when focusing solely on the classification task, the results are not optimal, and could be surely improved.



(a) AUC for DS 16.6.4 (b) AUC for DS 16.2.2

Figure 4.4: The images represent the AUC during the training of BRITS model on datasets with 16 variables.

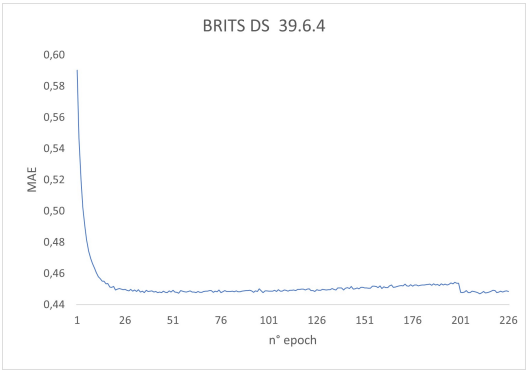
The results in terms of average loss for the two datasets with 39 variables are shown in Fig. 4.5. The DS 39.2.2 (class) dataset shows slight peaks in the loss curve, which may be due to the large batch size used, causing the weights to adjust significantly at the end of each epoch. However, this choice was made because, during previous trials with a batch size of 32, the adjustments were extremely slow, leading the model to plateau. Moreover in DS 39.2.2 without the classification part the learning rate was reduced at epoch 110. While in DS 39.2.2 with the classification the learning rate was reduced two times: at epoch 127 and 139.



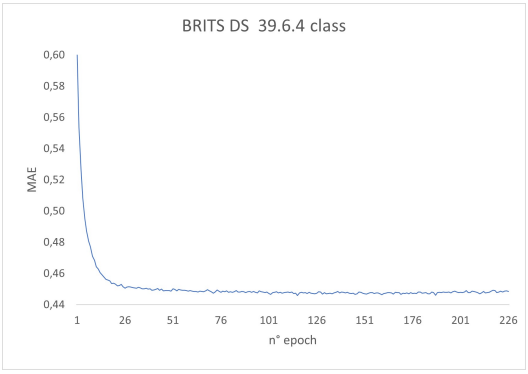
(a) Average Loss for DS 39.2.2 no class (b) Average Loss for DS 39.2.2 class

Figure 4.5: The images represent the average loss during the training of BRITS model on dataset with 39 variables. In Fig. 4.5a is showed the average loss when it is deleted from the model the classification part. While in Fig. 4.5b it is showed the average loss using the classification task, which has an impact on the total loss.

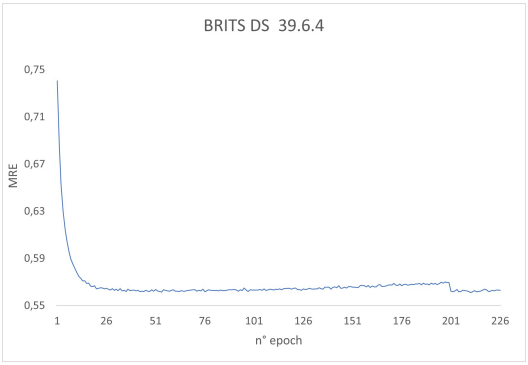
In Fig. 4.6 are presented the graphics of MAE and MRE over time. The results follow a similar trend to those obtained for the datasets with 16 variables, particularly in terms of the curve progression over the epochs.



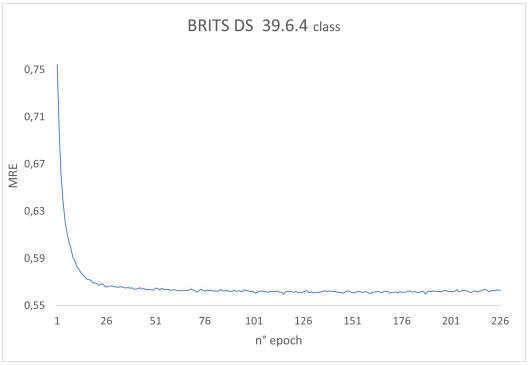
(a) MAE for DS 39.6.4 no class



(b) MAE for DS 39.6.4 class



(c) MRE for DS 39.6.4 no class



(d) MRE for DS 39.6.4 class

Figure 4.6: The images represent MAE and MRE during the training of BRITS model on dataset with 39 variables. The images shows the MAE and MRE during epochs with and without the classification part.

4.1.1 Comparison

Fig. 4.7 shows a comparison of the best results obtained for each dataset, with and without the classification component. The results demonstrate similar performances across all percentages of missing values, confirming that this model maintains high performance even in the presence of significant amounts of missing data.

	DS 16.2.2	DS 16.2.2 class	DS 16.6.4	DS 16.6.4 class	DS 39.2.2	DS 39.2.2 class	DS 39.6.4	DS 39.6.4 class
MAE %	34.27%	34.31%	43.12%	42.71%	32.91%	32.83%	44.74%	44.58%
MRE %	43.78%	43.83%	54.67%	54.16%	41.76%	41.67%	56.14%	55.94%
%NaN	34.70%		69.17%		58.85%		82.31%	

Figure 4.7: Comparison of best results obtained on each dataset; with and without classification using BRITS model.

Furthermore, these results are consistent with the state of the art. In "BRITS: Bidirectional Recurrent Imputation for Time Series" by Wei Cao, Dong Wang et al. [15] , the results obtained on Physionet 2012—a healthcare dataset briefly discussed in Chapter 2, with approximately 78% missing data—showed that MAE and MRE were 27.8% and 38.72%, respectively. Given the differences in computational resources and the fact that the type of missing data is not MNAR (the type where BRITS has demonstrated better performance), this variation in results is understandable.
However, one point worth discussing is that the performance does not improve as the percentage of missing data decreases. This could be attributed to the smaller amount of available information in such cases or possibly to the limited number of variables, which might be insufficient for the model to fully learn the underlying patterns.

Another observation is that the results with or without classification are not significantly different. This outcome was anticipated, as discussed earlier; however, the AUCs—which were predicted to be low—do not align with these expectations. This discrepancy could be attributed to the large number of patients (and thus labels) provided to the model, or possibly to a correlation between the patient records and extubation outcomes, regardless of the timeframes considered.
Additionally, the results in terms of AUC appear slightly better when more hours are taken into consideration (Fig 4.8), regardless of the percentage of NaN values in the dataset. The correlation between the hours considered and patient outcomes could be valuable information for future implementations. However, it is essential to recognize that more hours of sampling may bring the data closer to the extubation times of more patients, which could also explain the differences in

AUCs associated with varying sequence lengths.

	DS 16.2.2	DS 16.6.4	DS 39.2.2	DS 39.6.4
AUC [%]	71.61%	77.16%	71.82%	76.57%
NaN [%]	34.70%	69.17%	58.85%	82.31%

Figure 4.8: AUC results for BRITS model corresponding to the best MAE and MRE obtained.

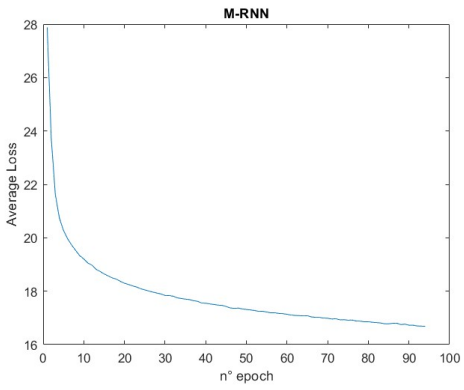
4.2 M-RNN

This model should be weaker than BRITS, because it does not consider the cross-correlation among variables. The parameters are set slightly different from BRITS. They are presented in Fig 4.9.

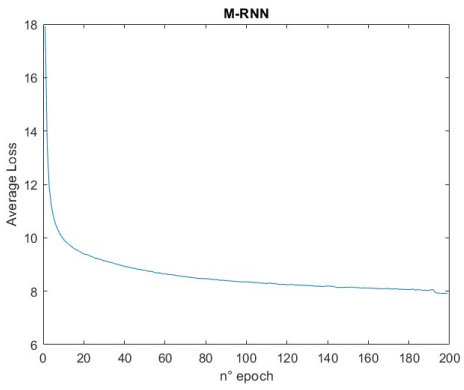
Batch size	64\128
Hidden Size	128
N° Epochs	100\200\300
Label Weight	1
Imputation Weight	1\3
Shuffle	True
Dropout	0.25
Learning Rate	10^(-3)

Figure 4.9: Parameters set for this model. Some rows have multiple numbers because the parameters have been set differently for the various cases analyzed. In particular, the imputation weight is different whether the classification part is used (3) or not (1).

In Fig. 4.10 is showed the average loss over time for the training with M-RNN, here as in BRITS, sometimes the performances reach a plateau so the epochs are stopped before they reach the number set.



(a) Average Loss for DS 39.6.4 no class

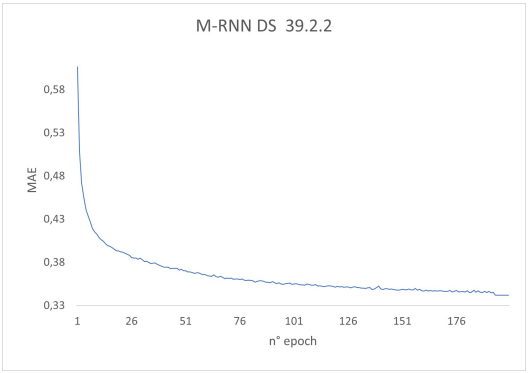


(b) Average Loss for DS 39.2.2 no class

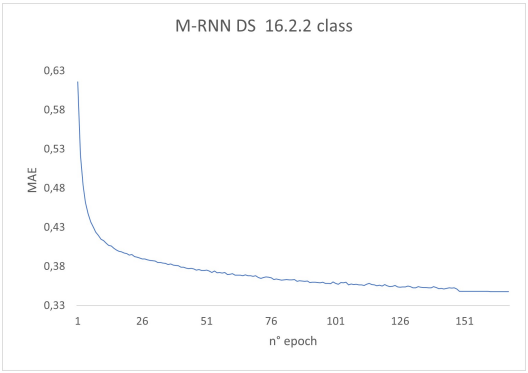
Figure 4.10: The images represent the average loss during the training of M-RNN model on dataset with 39 variables when the model does not use the classification.

In Fig. 4.11 there is a comparison of the MAE and MRE obtained on different datasets, each one with 39 variables, during M-RNN training. The graphs clearly demonstrate a stable convergence towards the best values for both metrics. This indicates that the model is capable of learning and improving its imputation

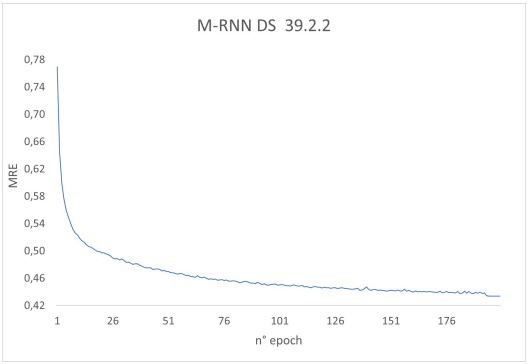
performance over time, with minimal oscillation or divergence during training. The stability of the convergence suggests that M-RNN is able to generalize across datasets with varying missing data percentages, consistently reducing both MAE and MRE as it progresses through training.



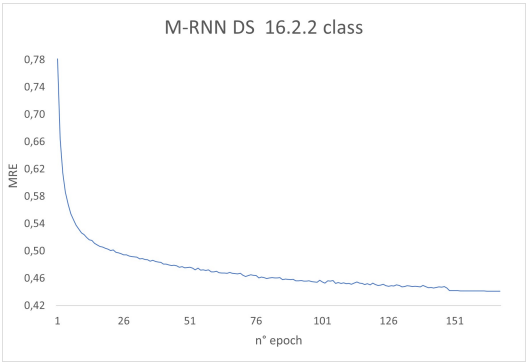
(a) MAE for DS 39.2.2 no class



(b) MAE for DS 39.2.2 class



(c) MRE for DS 39.2.2 no class

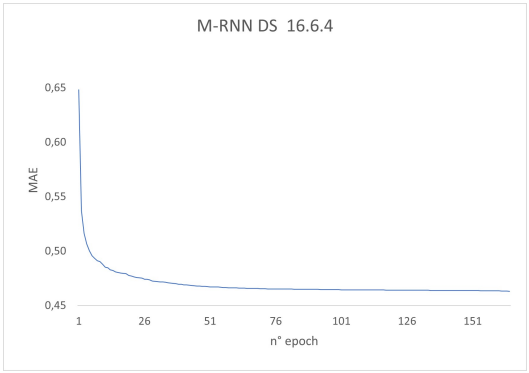


(d) MRE for DS 39.2.2 class

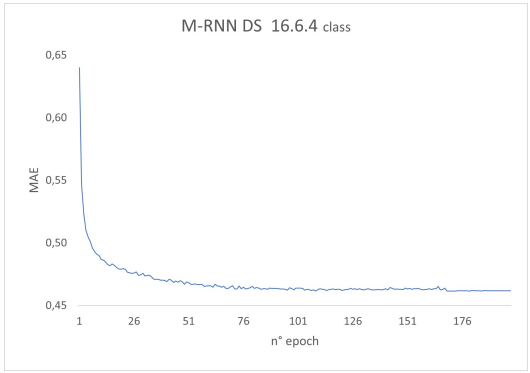
Figure 4.11: The images represent MAE and MRE during the training of M-RNN model on dataset with 39 variables. The images shows the MAE and MRE during epochs with and without the classification part.

In Fig. 4.12 there is a comparison of MAE and MRE obtained on different datasets with 16 variables, as it can be seen the values are slightly better than the ones presented in Fig 4.11.

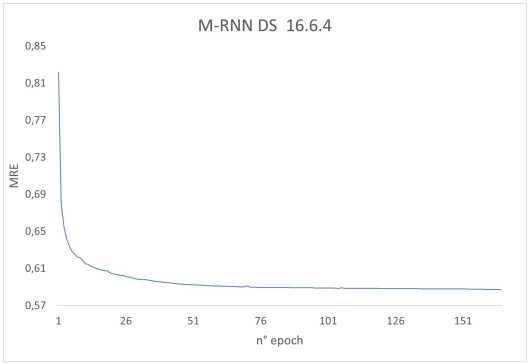
Fig 4.13 shows the AUC during training. The values are not stable, this could be due to the weight given to the classification which is small compared to the imputation one. These results could mean that M-RNN does not stabilize this part of the model when the weights are too unbalanced.



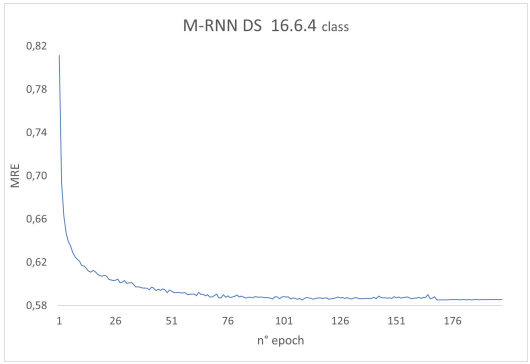
(a) MAE for DS 16.6.4 no class



(b) MAE for DS 16.6.4 class

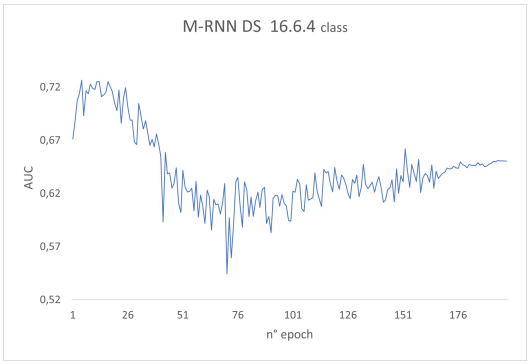


(c) MRE for DS 16.6.4 no class

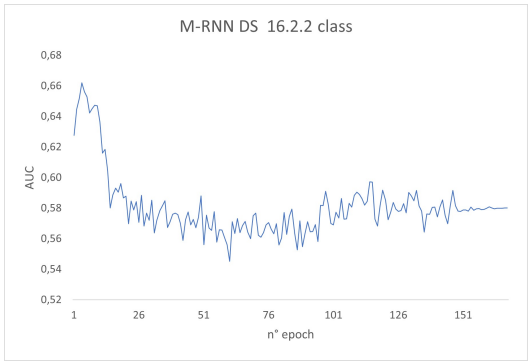


(d) MRE for DS 16.6.4 class

Figure 4.12: The images represent MAE and MRE during the training of M-RNN model on dataset with 16 variables. The images shows the MAE and MRE during epochs with and without the classification part.



(a) AUC for DS 16.6.4



(b) AUC for DS 39.2.2

Figure 4.13: The images represent the AUC during the training of M-RNN model on two different datasets.

4.2.1 Comparison

In Fig. 4.14, the results obtained in terms of MAE and MRE for the M-RNN model across each dataset are presented. In this case, the results align with the state of the art; specifically, in "BRITS: Bidirectional Recurrent Imputation for Time Series" by Wei Cao, Dong Wang et al. [15], the reported results on the Physionet dataset are 44.5% for MAE and 61.87% for MRE.

However, there is a more significant discrepancy between the results obtained for different percentages of missing values when compared to BRITS. This suggests that the complex structure of BRITS may provide better performance overall. In particular, performance declines when the percentage of missing values increases to 60%. The model appears to maintain relatively constant performance for missing values at 34% and 58%, but shows worse results as the percentage of missing values rises further. Overall this model seems to be a valid option for imputing missing values in dataset with a relatively small percentage of missing data, without the complexity of BRITS.

	DS 16.2.2	DS 16.2.2 class	DS 16.6.4	DS 16.6.4 class	DS 39.2.2	DS 39.2.2 class	DS 39.6.4	DS 39.6.4 class
MAE %	33.06%	33.21%	46.30%	46.15%	34.19%	34.77%	50.32%	50.12%
MRE %	42.24%	42.43%	58.70%	58.51%	43.39%	44.12%	63.15%	62.89%
%NaN	34.70%		69.17%		58.85%		82.31%	

Figure 4.14: Comparison of best results obtained on each dataset; with and without classification using M-RNN model.

In Fig. 4.15 a comparison of the AUCs obtained for each dataset is presented. Overall, the results in terms of classification are not satisfactory, as none of the percentages exceed 70%, which is generally considered an acceptable threshold in binary classification. In this model, the weight assigned to the imputation part was significantly higher, which may explain the poor performance in the classification task. However, despite not being ideal for a classification task, the results remain acceptable, as they do not fall below 50%.

	DS 16.2.2	DS 16.6.4	DS 39.2.2	DS 39.6.4
AUC [%]	60.47%	61.46%	58.00%	65.66%
NaN [%]	34.70%	69.17%	58.85%	82.31%

Figure 4.15: AUC results for M-RNN model corresponding to the best MAE and MRE obtained.

4.3 Discussion

In conclusion, the best performance overall in absolute terms was achieved by DS 39.2.2 using the BRITS model with incorporated the classification component. However, it is important to note that the results from other datasets also merit consideration, as they align closely with the current state of the art in the field. The disparities in performance between BRITS and the M-RNN model become more evident as the percentage of missing data increases. This observation suggests that taking into account the possible correlation among features could be important when imputing missing values. Additionally, this aspect appears to be more significant in scenarios where the proportion of missing values is high.

The findings indicate that there is no a substantial difference in terms of MAE and MRE between BRITS and M-RNN when the percentage of missing values is relatively low. Given the complexity of the BRITS model, the M-RNN could serve as an optimal alternative in such situations, offering a simpler yet effective approach.

In terms of AUC, the two models exhibits differences across each dataset and for every percentage of missing values considered. This suggest that while both models experience challenges due to the disparity between imputation and classification weights, BRITS is capable of adjusting to this imbalance and achieving relatively good performance in the classification task.

Moreover, when examining the relationship between time sampling and AUC during weaning, it appears that a larger time sampling is associated with improved AUC results. This correlation suggests that the records considered are highly relevant. The data indicates that a greater time sampling allows for more patients to be closer to extubation, reinforcing the idea that creating an effective dataset for predicting optimal weaning times could benefit from clustering patients based on their intubation timelines. Additionally, having more records of the same patient's data may enhance the model's ability to identify underlying patterns, which in turn could lead to improved classification performance. Further analysis should be conducted to establish which is the reason for the model to perform better in classification more records are presented, even if its working with the same number of labels.
Regarding the selected variables, the results appear to indicate a correlation among them not strong as it was expected. While the correlation among features may explain why BRITS outperformed M-RNN when a high percentage of missing values was present, it is also noteworthy that the two methods do not exhibit significant differences when dealing with lower percentages of missing data. Even when excluding classification outcomes, the results remain very similar.

Despite the initial hypothesis that the selected variables alone might not suffice for an effective classification task aimed at determining the optimal timing for

initiating the weaning process, the results support this notion. The apparent weak relationships among the variables appear inadequate for effective classification. This confirms the necessity of incorporating binary and static variables after processing time-series data to enhance the classification task.

In summary, the findings of this thesis underscore the importance of model selection and feature correlation in the context of missing data imputation and classification tasks. Future research could further explore the implications of these findings, particularly regarding how the integration of additional variables and the refinement of model architectures could lead to improved outcomes in predicting optimal weaning times. As the field continues to evolve, ongoing advancements in model development and data analysis will surely contribute to more accurate and effective clinical decision-making processes.

Bibliography

[1] Miguel Marino et al. *Missing data in primary care research: importance, implications and approaches.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8243609/.

[2] Pritha Bhandari. *Missing Data — Types, Explanation, & Imputation.* URL: https://www.scribbr.com/statistics/missing-data/.

[3] Ahmed Abulkhair. *Data Imputation Demystified — Time Series Data.* URL: https://medium.com/@aaabulkhair/data-imputation-demystified-time-series-data-69bc9c798cb7.

[4] Dhiraj Magare et al. *Imputation of missing data in time series by different computation methods in various data set applications.* URL: https://www.itm-conferences.org/articles/itmconf/pdf/2020/02/itmconf_icacc2020_03010.pdf.

[5] James Durbin and Siem Jan Koopman. *Time Series Analysis by State Space Methods (No. 38). Oxford University Press.* URL: https://www.researchgate.net/publication/227468262_Time_Series_Analysis_by_State_Space_Methods.

[6] Unknown Author. *What is ARIMA Modeling?* URL: https://www.mastersindatascience.org/learning/statistics-data-science/what-is-arima-modeling/.

[7] Jason Brownlee. *A Gentle Introduction to Expectation-Maximization (EM Algorithm).* URL: https://machinelearningmastery.com/expectation-maximization-em-algorithm/.

[8] Melissa J. Azur et al. *Multiple imputation by chained equations: what is it and how does it work?* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/.

[9] Bahram Jafrasteh et al. *Gaussian processes for missing value imputation.* URL: https://www.sciencedirect.com/science/article/pii/S0950705123003532.

[10] Shichao Zhang. *Nearest neighbor selection for iteratively kNN imputation.* URL: https://www.sciencedirect.com/science/article/abs/pii/S0164121212001586.

[11] Niklas Donges. *A Complete Guide to Recurrent Neural Networks (RNNs).* URL: https://builtin.com/data-science/recurrent-neural-networks-and-lstm.

[12] Anish Nama. *Understanding Gated Recurrent Unit (GRU) in Deep Learning.* URL: https://medium.com/@anishnama20/understanding-gated-recurrent-unit-gru-in-deep-learning-2e54923f3e2.

[13] Vincent Fortuin et al. *GP-VAE: Deep Probabilistic Time Series Imputation.* URL: https://proceedings.mlr.press/v108/fortuin20a/fortuin20a.pdf.

[14] Mayank Banoula. *Introduction to Long Short-Term Memory(LSTM).* URL: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm.

[15] Wei Cao et al. *BRITS: Bidirectional Recurrent Imputation for Time Series.* URL: https://papers.nips.cc/paper_files/paper/2018/file/734e6bfcd358e25ac1db0a4241b95651-Paper.pdf.

[16] Xiao Liu et al. *NAOMI: Non-Autoregressive Multiresolution Sequence Imputation.* URL: https://arxiv.org/pdf/1901.10946.

[17] Ping Zhang et al. *Identifying Sepsis Subphenotypes via Time-Aware Multi-Modal Auto-Encoder.* URL: https://www.pingzhang.net/files/KDD2020_TAME.pdf.

[18] Kejing Yin and William K. Cheung. *Context-Aware Imputation for Clinical Time Series.* URL: https://kejing.me/publications/2019_ICHI_CATSI_abs.pdf.

[19] Padmastuti Akella, Louis P. Voigt, and Sanjay Chawla. *To Wean or Not to Wean: A Practical Patient Focused Guide to Ventilator Weaning.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10329429/.

[20] Chung-Feng Liu et al. *An artificial intelligence system to predict the optimal timing for mechanical ventilation weaning for intensive care unit patients: A two-stage prediction approach.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9715756/.

[21] Bronagh Blackwood et al. *Protocolized versus non-protocolized weaning for reducing the duration of mechanical ventilation in critically ill adult patients.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6517015/.

[22] Mario Fadila, Venkat Rajasurya, and Hariharan Regunath. *Ventilator Weaning.* URL: https://www.ncbi.nlm.nih.gov/books/NBK430712/#:~:text=Prolonged%20mechanical%20ventilation%20increases%20the,long%2Dterm%20care%20facility%20discharge.

[23] IBM Data and AI Team. *AI vs. machine learning vs. deep learning vs. neural networks: What's the difference?* URL: https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks#:~:text=Deep%20learning%20is%20a%20subfield,must%20have%20more%20than%20three.

[24] Statista Research Deparrment. *Digital health - Statistics & Facts.* URL: https://www.statista.com/topics/2409/digital-health/#topicOverview.

[25] A.E.W. Johnson et al. *MIMIC-IV, a freely accessible electronic health record dataset.* URL: https://doi.org/10.1038/s41597-022-01899-x.

[26] Wikipedia. *SAPS II.* URL: https://en.wikipedia.org/wiki/SAPS_II.

[27] Michael B. Keller et al. *Preintubation Sequential Organ Failure Assessment Score for Predicting COVID-19 Mortality: External Validation Using Electronic Health Record From 86 U.S. Healthcare Systems to Appraise Current Ventilator Triage Algorithms.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9196924/.

[28] Maksims Kazijevs and Manar D. Samad. *Deep imputation of missing values in time series health data: A review with benchmarking.*

[29] Dingwen Li et al. *Integrating Static and Time-Series Data in Deep Recurrent Models for Oncology Early Warning Systems.* URL: https://www.cs.wustl.edu/~lu/papers/cikm21.pdf.

[30] Alvaro Sanabria et al. *Prediction of prolonged mechanical ventilation in patients in the intensive care unit A cohort study.* URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4002035/.

[31] Yurim Lee et al. *Multi-View Integrative Attention-Based Deep Representation Learning for Irregular Clinical Time-Series Data.* URL: https://ieeexplore.ieee.org/document/9769928.