Challenges in Detecting Event Patterns in PICU Data

A GRU and Clustering-Based Study

Sine Bijsterveld

Supervisors UU: Arno Siebes & Ad Feelders

Supervisors UMC/WKZ: Ruben Zoodsma & Lex van Loon

A thesis presented for the degree of
Master in Data Science

Utrecht University

Netherlands

2024

Abstract

This thesis focuses on the use of Gated Recurrent Units (GRUs) and clustering techniques to improve alarm management systems in pediatric intensive care units (PICUs). The study aimed to identify event patterns which included an analysis of physiological data from patients in the Pediatric Intensive Care Unit (PICU), implementing GRU models for predicting events, and hierarchical agglomerative clustering (HAC) to group similiar events.

The results found that although the GRU model showed a high level of accuracy in identifying non-events, it had significant challenges with accurately detecting the real events. This resulted in a large number of false positives. The hierarchical agglomerative clustering (HAC) results showed promising results. Cluster 1, including 79% of all events mostly consisted of stable events with small parameter fluctuations. This suggests that these patterns were incorrectly classified as event and are not actual clinical or artifact events. On the other hand, Cluster 2, which consisted of 34 events, displayed more fluctuations and rapid changes in the vital signs. This indicates that the events are actual clinical events or artifacts. The outlier clusters consist of events that also showed many fluctuations, indicating a real event. However, the duration of the outliers is much longer. This could indicate longer periods of a patient in distress.

The results indicate that the GRU model in combination with the HAC model can effectively extract and identify events from PICU data. However, the accuracy and dependability of the results are influenced by possible inaccurate labeling by a non-expert, constraints of the computational resources, reduction of data size, and model limitations.

The study highlights the potential and limitations of using machine learning to enhance alarm management in PICUs. However, improvement and more research are needed to develop more effective systems for recognizing events, and to reducing alarm fatigue and improving patient safety.

"A digital health ecosystem that provides high-quality, personalized, equitable care to all who need it is achievable and worthy of our best individual and collective efforts." – National Academy of Medicine [1]

Contents

List of Figures

List of Tables

1 Introduction

Alarm fatigue is a pervasive issue in pediatric intensive care units (PICUs), significantly impacting patient safety and the efficiency of healthcare providers. Alarm fatigue occurs when clinicians become desensitized to alarm signals due to the overwhelming frequency and often inappropriate nature of these alarms. This phenomenon can lead to delayed responses, missed alarms, and ultimately, adverse patient outcomes [2]. The PICU environment, characterized by its patients' critical and often unstable conditions, is particularly susceptible to this problem [3].

Patients in PICUs, especially those suffering from heart disease or other severe conditions, exhibit considerable physiological fluctuations. These fluctuations can result from the illness itself, medical interventions, or various external factors such as the presence of family members, interactions with healthcare staff, pain, movement, or coughing. For instance, a patient's heart rate might spike due to a painful procedure or the anxiety of seeing a doctor, and blood pressure might temporarily increase following the administration of epinephrine [4]. Such variability in physiological parameters can trigger numerous alarms, many of which are false or clinically insignificant, contributing to alarm fatigue [5].

Addressing alarm fatigue is crucial for several reasons. Firstly, it directly affects patient safety. Critical alarms might be overlooked or ignored in an environment where alarms are constant, delaying necessary interventions [6]. Secondly, it impacts the mental and emotional well-being of healthcare providers. Continuous exposure to frequent alarms can lead to increased stress, reduced job satisfaction, and burnout among nurses and doctors [2]. Thirdly, the overall efficiency and effectiveness of healthcare delivery are compromised. Time spent responding to false alarms could be better utilized in direct patient care, improving health outcomes [7].

Synthetic patient data was generated for research and development purposes to improve alarm management and reduce alarm fatigue. However, this synthetic data typically lacks the complexity and noise found in real patient data. Real patient data contains essential fluctuations reflecting the dynamic nature of patients' conditions and their responses to various stimuli and treatments [8]. This complexity is particularly pronounced in patients with heart conditions, who might experience sudden changes in heart rate and blood pressure due to medical interventions or changes in their physical or emotional state.

At the Wilhelmina Children's Hospital in the Netherlands, they work on enhancing the alarm management system from the PICU by leveraging advanced data analytics and machine learning techniques. One of the critical steps in this initiative involves identifying and extracting specific segments of patient data that correspond to significant clinical events, such as the administration of oxygen or epinephrine. These events are known to

cause notable short-term changes in physiological parameters, making them pivotal for refining alarm systems.

To address the challenge of detecting these events within the data, this research proposes using Gated Recurrent Units (GRUs). GRUs are particularly well-suited for this task due to their high accuracy but manageable computational weight. By training a GRU model on annotated patient data, the system can learn to detect when an event is occurring automatically and find different event patterns, thereby improving alarm systems' precision and reducing false alarms' incidence [9]. The ultimate goal of this research is to leverage GRUs to analyze real patient data, capturing the nuances and fluctuations characteristic of patients from the PICU. By detecting event patterns in the patient data, this study will provide extra knowledge in exploring strategies to visualize the event patterns. It also aims to set a small step in the direction of reducing alarm fatigue, thereby enhancing patient safety and improving outcomes for critically ill children.

The literature review that follows in section 2 provides an overview of existing research and strategies aimed at mitigating alarm fatigue. It highlights the potential of machine learning and data-driven approaches and underscores the importance of using real patient data to train models that can effectively manage alarms in a clinical setting. Next, Section 3 will elaborate on the dataset and the missing data that is utilized in this research. In Section 4, the approach of data labelling, handling the dataset, implementing the GRU, clustering, and visualization is elaborately discussed. Following, we propose the main findings in Section 5. Then, we discuss the findings and the strengths and limitations of this study in Section 6. Finally, we conclude the work with final remarks in Section 7.

2 Literature Review

In this section, we present the literature background on alarm fatigue and specifically focus on machine learning models to combat this significant problem. Section 2.1 explains what alarm fatigue involves. After that, the strategies that have been proposed and evaluated to combat alarm fatigue are discussed in Section 2.2. Besides, we highlight some studies and their techniques to overcome alarm fatigue. In Section 2.3, we look at the use of Gated Recurrent Units (GRUs) in combination with time series analysis as a possible new direction. Lastly, a literature review summary will be in Section 2.4.

2.1 Alarm fatigue

Alarm fatigue, defined as the desensitization or decreased responsiveness of healthcare providers to alarms due to excessive or inappropriate alarm signals, poses a significant challenge in patient care settings [2]. This issue is particularly acute in pediatric intensive care units (PICUs), where alarms are prevalent due to the critical condition of the patients. Clinical staff, who continuously monitor patients, are especially vulnerable to alarm fatigue [3]. The high volume of alarms, with 72% to 99% being false, exacerbates this problem. Alarm fatigue has serious consequences, including missed alarms and patient deaths, leading to a Joint Commission National Patient Safety Goal since 2014. Numerous patient safety initiatives and regulations have subsequently focused on this issue [2, 4]. This literature review aims to explore the existing research on strategies to mitigate alarm fatigue, with a specific emphasis on machine learning methods, including using Gated Recurrent Units(GRUs) for time series analysis.

2.2 Research on Strategies to Combat Alarm Fatigue

Addressing alarm fatigue has been a major focus of healthcare research globally. Various strategies have been proposed and evaluated to mitigate its adverse effects on patient care and safety. A comprehensive literature review reveals several key studies that have significantly contributed to understanding this complex phenomenon and offered potential solutions.

Chromik et al. [8] conducted a systematic review of efforts to develop IT systems to reduce alarm fatigue. The study focused on avoiding false alarms, predicting patient deterioration, and improving alarm presentation. Most publications targeted heart rate or arrhythmia alarms, emphasizing software solutions and tree-based statistical models. IT-based solutions, particularly those addressing technically false alarms, were found to be effective in reducing alarm fatigue in ICUs. Venkateswaran et al. [6] supported these findings, implementing a systematic alarm management pilot that reduced alarm fatigue at a tertiary care center.

Sendelbach and Funk [2] explored quality improvement projects demonstrating that strategies such as daily electrocardiogram electrode changes, proper skin preparation, education, and customization of alarm parameters could decrease the number of false alarms. However, these strategies require rigorous clinical trials to ensure they reduce the alarm burden without compromising patient safety. Hussain et al. [10] proposed more personalized alarm parameters, using fast-and-frugal heuristics like prioritizing alarms based on ventilator presence, the number of intravenous drips, and the number of medications to reduce alarm fatigue.

Advanced analytics and machine learning techniques have also shown potential in optimizing alarm systems and reducing alarm burden. Pater et al. [11] employed machine learning algorithms to analyze alarm patterns in a pediatric hospital setting, informing the development of data-driven strategies for alarm customization and reduction of non-actionable alarms.

Wang et al. [12], Hyland et al. [13] developed machine learning models that achieved high accuracy in false alarm detection and early prediction of circulatory failure. Wang et al. [12] used a multivariate approach to identify false alarms generated by ICU bedside monitors, capturing characteristics from both arterial blood pressure (ABP) and electrocardiogram (ECG) signals. This approach mitigates the imprecision caused by existing heartbeat/peak detection algorithms and does not require separate techniques for different types of alarms. Hyland et al. [13] developed an early-warning system that integrates measurements from multiple organ systems using a high-resolution database. Based on an array of demographic, physiological, and clinical information, their algorithm can predict circulatory failure hours in advance.

These studies highlight the importance of a multifaceted approach to combat alarm fatigue, particularly data-driven methods. Machine learning techniques have shown promising results, achieving high accuracy. By employing these techniques, healthcare organizations, especially ICUs, can effectively address alarm fatigue and enhance patient safety across diverse care settings.

2.3 LSTM and GRU for multivariate time series data

One model that shows very promising results is a Long-Short-Term-Memory (LSTM). Multiple studies have shown the effectiveness of a LSTM in analyzing medical time series data. For example, Park et al. [14] introduced the LSTM-MDN-ATTN model, which uses an attention mechanism to predict future values in medical data. Another study looked into the ability of LSTM networks to recognize patterns in clinical measurements. Therefore, they used data from ICU patients and achieved superior performance. Especially comparing the LSTM to other models [15]. These studies highlight the potential of LSTM models in analyzing multivariate medical time series data. In 2014, Cho et al. [16]

introduced a new recurrent network, the Gated Recurrent Units (GRUs). The GRU is a recurrent neural network architecture that is much like a LSTM but has a gating mechanism to pick input or forget specific features. However, it does not have a context vector or output gate, leading to fewer parameters compared to a LSTM [17]. A comparison of LSTM and GRU in multiple applications shows that GRU is generally faster and more efficient than LSTMS [18]. Considering the computational constraints for this research, we will utilize Gated Recurrent Units (GRUs).

2.4 Conclusion

Alarm fatigue remains a significant issue in healthcare, particularly in critical care environments like PICUs. The high volume of false alarms contributes to desensitization among clinical staff, potentially compromising patient safety. This literature review highlighted various strategies to mitigate alarm fatigue, from IT-based solutions and personalized alarm parameters to advanced analytics and machine learning techniques. The integration of time series analysis with Gated Recurrent Units (GRUs) presents a promising direction for future research in event pattern detection. Implementing this innovative model aims to detect event patterns and ultimately improve healthcare systems by reducing alarm fatigue, enhancing patient safety, and improving overall care quality. Continued exploration and rigorous testing of multiple models and approaches are essential to developing effective, sustainable solutions to this pervasive problem.

3 Data

3.1 Data Set

This research focused on babies under one year old with congenital heart disease (cCHD) admitted to the Pediatric Intensive Care Unit (PICU) at the University Medical Centre Utrecht from 2002 to 2018. The dataset obtained from this was quite large, with a total of 12,025,453 rows. Every patient had gotten a pseudo_id. In total there where 68 patients in the dataset. The dataset included several key physiological parameters recorded every second. The parameters that were used can be seen in Table 1.

Table 1: Description of Parameters

Parameter	**Description**
mon_etco2	Measures the level of carbon dioxide at the end of exhalation.
mon_hr	Indicates heart rate, a fundamental measure of cardiac activity.
mon_ibp_mean	Mean arterial blood pressure, for understanding cardiovascular stability.
mon_nibp_mean	Non-invasive mean blood pressure.
mon_sat	Oxygen saturation in the blood, for monitoring respiratory health.
nirs_l	Brain oxygenation on the left side.
nirs_r	Brain oxygenation on the right side.

3.2 Missing data

The dataset consisted of many periods of NaNs for several patients. One of the reasons for missing data is the size of the head of the infant. When the size is very small, there is not enough space for the electrode stickers to measure the NIRS. This kind of missing data is called Missing at Random (MAR). Besides, for some infants, there are periods when only NIRS data is available due to another research study. The loss of this data can be missing not at random (MNAR) or missing completely at random (MCAR). Table 2 shows the number of missing rows for each parameter. It's important to consider that the total number of data points is 228 million, so the number of missing rows should be viewed with this in mind to see it from the right perspective. Significantly, the missing data of the average of the non-invasive blood pressure (mon_nibp_mean) is very high compared to the others. The specific reasons are unknown. However, it's likely MNAR since the other parameters don't suffer that much from missing data. Table 3 shows the combination of missing parameters for each patient. To calculate the combinations of missing data patterns, a parameter was considered missing only if there were no measurements for that parameter for the entire data for that particular patient. For example there are 36 patients that are missing the parameters: mon_etco2, mon_hr, mon_ibp_mean,

mon_nibp_mean, mon_sat, nirs_r.

Different techniques for the missing data were considered to make the dataset manageable. At first, forward-backward fill was considered. However, due to the high computation, this wasn't possible with the used virtual machine. Another option considered was taking the mean of the data per patient. However, this could create steep slopes in the graphs when the missing data is no longer missing. For example, if the mean value is six and the first actual measurement after the missing data is 15, this sharp increase could be mistaken for an event. Therefore, a complete case analysis was used, where all instances of missing data (NaNs) were deleted to ensure only complete cases were included. The data was then downsampled to one measurement every 3 seconds. The downsampling made calculations with the data much faster and easier to work with. After these steps, the dataset consisted of 25 patients and 348,979 rows.

Table 2: Total number of NaNs in the dataset

Parameter	**NaNs**
mon_etco2	5,526,843
mon_hr	4,517,294
mon_ibp_mean	5,448,108
mon_nibp_mean	9,649,067
mon_rr	4,714,957
mon_sat	4,686,772
nirs_l	1,356,902
nirs_r	1,419,864

Table 3: Combination of missing parameters per patient

Missing Combination	**Value**
(mon_etco2, mon_hr, mon_ibp_mean, mon_nibp_mean)	1
(mon_etco2, mon_hr, mon_ibp_mean, mon_nibp_mean, mon_sat)	6
(mon_etco2, mon_hr, mon_ibp_mean, mon_sat, nirs_r)	9
(mon_etco2, mon_hr, mon_ibp_mean, mon_nibp_mean, mon_sat, nirs_r)	36
(mon_etco2, mon_hr, mon_ibp_mean, mon_sat, nirs_l, nirs_r)	9
(mon_etco2, mon_ibp_mean, mon_sat, nirs_l, nirs_r)	2
(mon_hr, mon_ibp_mean, mon_nibp_mean, mon_sat)	1
(mon_hr, mon_ibp_mean, mon_nibp_mean, mon_sat, nirs_l, nirs_r)	1
(mon_ibp_mean, mon_sat, nirs_l, nirs_r)	1
(mon_ibp_mean, nirs_l, nirs_r)	1
(nirs_l, nirs_r)	1

4 Methodology

In this section, we elaborate on the methodology. Section 4.1 covers data preprocessing, including initial steps, labeling data, and dataset handling. The training and the prediction with the GRU model will be discussed in Section 4.2. The clustering of events is explained in Section 4.3, where we discuss the separation of events, Dynamic Time Warping, determining the optimal number of clusters, clustering itself, and the visualizations of the clusters. Lastly, Section 4.4 provides an overview of the implementation and tools used throughout the study.

4.1 Data Preprocessing

4.1.1 Initial Steps

The initial data preprocessing phase consisted of importing the dataset and thoroughly examining the column names, data types, and beginning statistics to fully understand the data structure. The ‘pat_datetime‘ column, which indicates the timestamp of each record, was transformed into a DateTime object. This transformation was essential to time-based processes and provided accurate temporal data synchronization. The dataset was organized by ‘pat_datetime‘ and ‘pseudo_id‘, ensuring that the records for each patient were arranged in chronological order.

4.1.2 Labeling Data

The first ten patients’ data were manually reviewed and labeled to train the model. This initial manual labeling process involved several detailed steps to ensure the creation of a well-annotated initial training set:

1. *Visualization of Vital Parameters* First, plots were created for each patient’s vital parameters over time. A time interval of 6 hours was chosen for these plots to provide a clear visualization of how the parameters changed over a manageable period. This interval allowed for identifying trends and patterns that might indicate abnormal events.

2. *Identification of Abnormalities* Each graph was extensively studied, and any abnormalities were marked. Abnormalities could include sudden spikes or drops in vital parameters, or other deviations from the expected patterns. These marked abnormalities were critical for identifying periods of interest that required further analysis.

3. *Labeling of Events* The abnormal time frames identified in the previous step, referred to as “events,” were then added to the dataset as labeled events. An event

was annotated as '1,' while a non-event was annotated as '0.' This binary labeling ensured a clear distinction between periods of non-events and events.

Manual labeling brings valuable human insight to the data, capturing small nuances and contextual information that automated systems might overlook. This can significantly enhance the accuracy and relevance of the labeled data. Besides, it can help train a model when no labels or pre-trained models are available beforehand. However, the process is time-consuming and depends heavily on individual judgment, which can introduce subjectivity and bias into the dataset. In this study, the manual labeling was performed by a non-expert, which could introduce personal biases and inconsistencies. Unlike an expert clinician, the interpretations were based on a non-expert understanding, which might have led to errors or mislabeling. This subjectivity can impact the model's ability to generalize and perform well. Therefore, while manual labeling provides significant benefits through human insight, its drawbacks must be acknowledged, including the potential for bias and the impact on model performance. Understanding these limitations is crucial for interpreting the results.

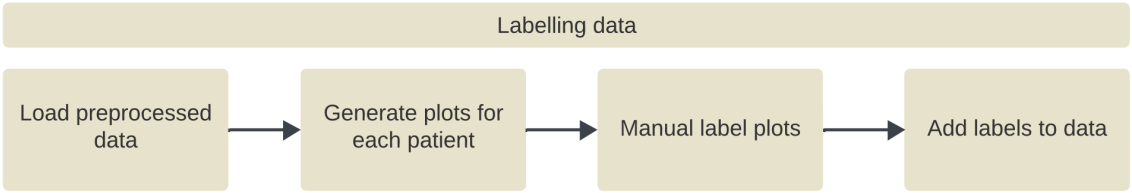


Figure 1: Flow chart: Data labeling

4.1.3 Data set handling

Normalization Normalization ensured that all physiological parameters contributed equally to the model. Therefore, the StandardScaler from the sci-kit-learn library was used. The scaler was also saved to inverse transform the data after being used in the model. Normalization is important to prevent features with higher scales from having too much influence on the model. Therefore, it improves the model performance and convergence rates.

Data splitting Patients were manually selected and divided into training and validation sets to account for each patient's different amounts of data. This provided a more equal distribution of the data. With the manual selection method, the length and comprehensiveness of the data for each patient were considered to create well-balanced and representative training and validation sets. This approach ensured that the model was trained on a more balanced dataset while maintaining the accuracy and reliability of the validation process. This approach allows a more regulated setup and, given the relatively limited size of the training data, a more controlled environment. Nevertheless, this task requires a significant amount of manual effort and must be executed cautiously to avoid any potential bias in the selection process.

Preparing for GRU A GRU model will be used to predict the events for this task. It looks at the data point and the history of the data point it needs to predict. Therefore, we need to specify how many rows the GRU takes into account when predicting. For this, the 2-dimensional data frame, the rows that the GRU needs for the prediction, were added as an extra dimension. The extra dimension was of size 300. With a step size of 300 * 3 seconds (due to downsampling), the 900 seconds before were used for the prediction.

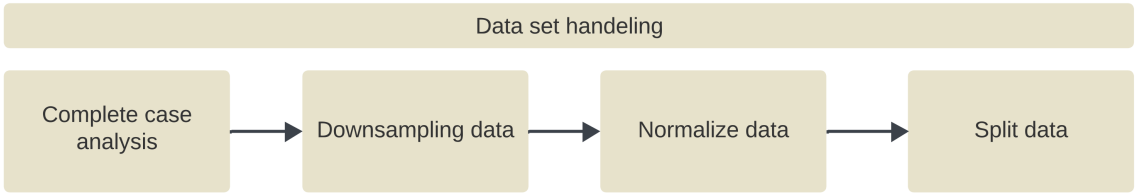


Figure 2: Flow chart: Dataset handeling

4.2 Model Development

4.2.1 Model training

The choice of a Gated Recurrent Unit (GRU) neural network was based on its faster training speed and efficiency than Long Short-Term Memory (LSTM) networks. This is because GRUs have a simpler structure than LSTMs. Therefore, the training of a GRU model goes much quicker than an LSTMS model's. GRUs have a simple structure; however, they still display a high effectiveness in sequential tasks. They even almost reached the performance level of LSTMs. Considering the limitations of our computational resources, the GRU model is a suitable option. Furthermore, GRUs show a lower sensitivity to gradient-related problems. However, its less complex structure could increase the possibility of overfitting, which should be considered. To address this issue, carefully choosing the hyperparameters for optimization is crucial.

4.2.2 Parameters

The GRU's hyperparameters were adjusted to optimize the model's performance.

Layers : The model was designed with four GRU layers and set the GRU layer units at 50. The GRU layers are developed to return the sequences to the next layers. When implementing multiple GRU layers, the model will stack the layers and will be able to find the patterns in the data. To prevent overfitting and improve generalization, a dropout layer was implemented for every layer of the GRU, and it was set at a rate of 0.2. This will randomly set a few input units to zero during the training. A rate of 0.2 was chosen to balance the risk of underfitting and overfitting, After the GRU layers, a dense layer was implemented. The output of this layer is the predicted value, a continuous variable that gives the probability of an event happening.

Adam optimizer The Adam optimizer controls sparse gradients and adjusts the learning rate throughout the training process. The learning rate will be reduced when the model doesn't achieve better results through the epoch.

Batch size The batch size was set to 64 to balance the computing efficiency and the stability of gradient updates.

Mean squared error Although Binary Cross-Entropy is the standard loss function for binary classification tasks, we used Mean Squared Error (MSE) for this study. MSE measures the average squared difference between the predicted probabilities and the actual binary labels (0 or 1). While MSE is typically used in regression tasks, it can also be applied to binary classification when treating the model's output as a probability.

Class weights Given the dataset's imbalance, class weights were computed to ensure the model paid sufficient attention to the minority class. This adjustment helped the model to learn patterns associated with rare events more effectively.

Epochs The model was trained for up to 100 epochs, with early stopping applied to halt training if the validation loss did not improve for 10 consecutive epochs. The model was sufficiently trained after around 26 epochs. By setting the epoch at a high number, the epoch would definitely be enough to prevent the model from running again due to not being trained sufficiently.

Early stopping Early stopping was used to prevent overfitting and reduce unnecessary computations by monitoring validation loss and restoring the best weights. The training process was tracked using accuracy and loss metrics for both training and validation sets. These metrics provided insights into the model's learning progress and its ability to generalize to unseen data, with validation accuracy being crucial for assessing performance on non-training data.

Implementing these settings allows the GRU to perform its best on time series data to predict events.

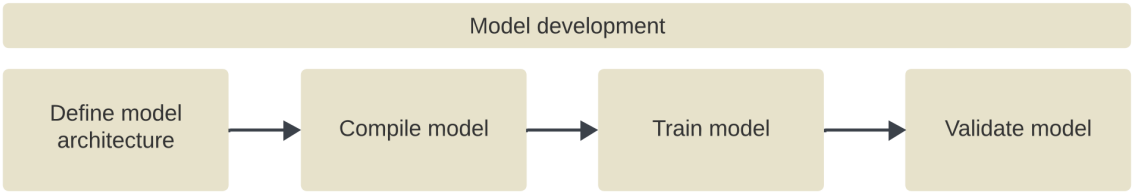


Figure 3: Flow chart: Model Development

4.2.3 Model prediction

To make the predictions, the data needed to be prepared to be suitable for the model. The prediction data was prepared in the same way as the training and the validation data to ensure that the prediction data was suitable and comparable to the training data. The data had to be organized into a three-dimensional dataframe to accomplish this.

Therefore, the step size was added as an extra dimension. The step size was set at 300, indicating that 900 seconds of data(15 minutes) were considered for each prediction.

After preparing the prediction data, the model was applied to make predictions on the unlabeled data to predict the events. The predictions made by the model were continuous numbers that represented the likelihood of an event. Next, a threshold of 0.5 was implemented so that these predictions are binary. Underneath the threshold the data point was given a 0 indicating a non-event and on the threshold or above a 1 was given indicating an event.

The data was converted back into a two-dimensional data frame to use for additional analysis. To do this, the three-dimensional prediction data had to be flattened and then normalized with the same transformer used for normalization. After that, the columns corresponding to each feature were reinserted, ensuring that each feature was correctly identified. Following these steps, the data was ready for analysis.

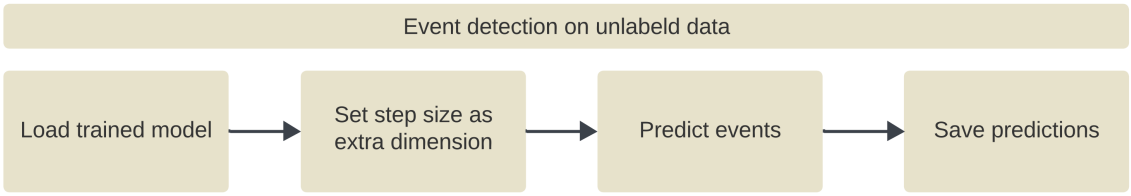


Figure 4: Flow chart: Event Predictions

4.3 Clustering and visualization of Events

4.3.1 Clustering method

This section will elaborate on the steps to obtain the clusters. First, the data was separated into cluster segments. Next, the distance matrix was calculated using Dynamic Time Warping (DTW). Multiple clustering techniques were approached, including Kmeans, DBSCAN, and OPTICS, but hierarchical agglomerative clustering (HAC) produced the best results. The following sections will discuss these steps in more detail.

1. *Separate Events* The data was initially divided into separate events by determining their beginning and ending positions. Therefore, the 'event' labels in the dataset were used to sort all the events as separate event segments. This was done on the combined dataset of the manually labeled data and the labeled data from GRU.

2. *Dynamic Time Warping* Next, the DTW algorithm was applied to calculate the distances between event segments. This method is highly efficient in measuring the similarity between the event segments. Therefore, it arranges the sequences in a way that minimizes the distance between them. With this, it can compare the sequences that are not perfectly time-aligned and is especially good to use for time

series data with different time lengths. Therefore, the distance matrix provided pairwise distance measures between all event segments.

3. *Determining the Optimal Number of Clusters* A hierarchical agglomerative clustering (HAC) without specifying the number was used to determine the optimal number of clustering. When not specifying the number of clusters, a distance matrix should be used as input. Therefore, the distance matrix made with DTW was used as input. Following this, a dendrogram was made to inspect the best cut for the clusters. After inspecting the dendrogram, the cutting point was set at a distance of 3000. This resulted in two clusters and 5 outlier events. The elbow method was also used to determine the best number of clusters and calculated the square distances for different cluster numbers. The elbow point is when the total of the squared clusters slows down. This is seen as the ideal number of clusters, which balances the complexity and the fit of the model. The elbow point was seen at 2 clusters, similar to the observation in the dendrogram.

4. *Clustering* Following, the event segments were then again put in the HAC with a cutting point of 3000. The advantages of this clustering method are not having to determine the number of clusters in advance. Also the model is flexible with the variety of densities of the data.

Using these methods, clusters were made to examine the different event patterns. Analyzing the clusters gave insight into this method's effectiveness.

4.3.2 Visualization and Analysis

The clustered events were visualized in time series graphs, plotting the medical parameters alongside predicted event clusters. Therefore, the cluster results of HAC were used as input. Visualizations were done with the matplotlib library. To display the different medical parameters, a legenda and colormap were used. Besides, extra parameters were used to improve the graphs' readability. The output of the clustering was very diverse, therefore subplots for every cluster were used for visualization and comparison. The goal of the visualizations is to see if this technique works to detect the events and, secondly, to see if there is a clear difference between the variety of events. The visualizations of the clusters will be discussed in Section 5 and can be seen in the Appendix.

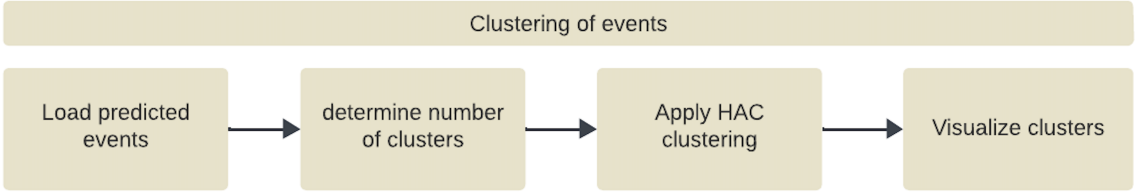


Figure 5: Flow chart: Clustering events

4.4 Implementation and Tools

The entire process, from data prepossessing to model training, event prediction, and evaluation, was implemented in Python using different libraries. Underneath is a small description of the used libraries.

- *Pandas*: Efficient data manipulation and analysis, easily handling large datasets.
- *Numpy*: Provides support for large, multi-dimensional arrays and matrices and a collection of mathematical functions to operate on these arrays.
- *Scikit-learn*: A comprehensive suite of machine learning tools, facilitating tasks like data preprocessing, model training, and evaluation. Specific tools used include *HAC* for clustering and *classification_report* and *confusion_matrix* for model evaluation.
- *TensorFlow/Keras*: Powerful frameworks for building and training deep learning models, offering flexibility and scalability. Specific components used include *GRU* for recurrent neural networks. Also, the hyperparameters are from this library.
- *tqdm*: Simple and intuitive progress bars enhance user experience during long-running processes.
- *joblib*: Efficient serialization of large Python objects and provides utilities for pipelining and parallelizing computations.
- *gzip*: A module for reading and writing GNU zip files, useful for data compression and decompression.
- *matplotlib*: A plotting library for creating static, animated, and interactive visualizations in Python.
- *fastdtw*: An approximate Dynamic Time Warping (DTW) algorithm that reduces computational complexity.
- *scipy*: Provides a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. Specifically, *spatial.distance* was used for the distance matrix and the dendrogram.

5 Results

5.1 Model accuracy

5.1.1 Classification Report

The classification report is shown in Table 4. The model shows high precision for the negative class (0.0), indicating it is almost always correct when predicting a negative class. However, it struggles with the positive class (1.0), which has a precision of only 0.19, implying many false positives. The recall for the positive class is high (0.99), showing that the model identifies most positive instances but at the cost of precision. The overall accuracy is 0.84, mainly due to the high performance of the negative class.

Table 4: Classification Report

Class	Precision	Recall	F1-Score	Support
0.0	1.00	0.83	0.91	38994
1.0	0.19	0.99	0.32	1564
Accuracy		0.84		
Macro avg	0.60	0.91	0.61	40558
Weighted avg	0.97	0.84	0.89	40558

5.1.2 Confusion Matrix

The confusion matrix in Table 5 displays the model's performance. The model accurately identifies most negative instances (32414 out of 38994). For the positive class, it correctly identifies 1554 instances but misclassifies 10 as negative. This imbalance highlights the need for improved precision for the event class.

Table 5: Confusion Matrix

	Predicted: 0	Predicted: 1
Actual: 0	32414	6580
Actual: 1	10	1554

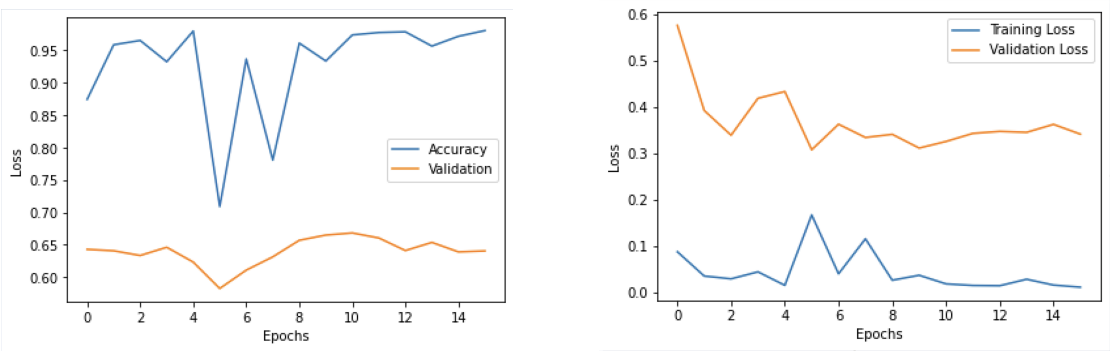


Figure 6: Accuracy matrices

5.2 Cluster Analysis

5.2.1 Number of Clusters

Elbow Plot The elbow plot, shown in figure 7, helps determine the ideal number of clusters by plotting the sum of squared distances against the number of clusters. The "elbow" point shows the location where the rate of decrease significantly slows down, indicating the optimal number of clusters. In this instance, the best number of clusters was found to be 2. This aligns with the results obtained from the hierarchical clustering, which will be discussed in the next section.

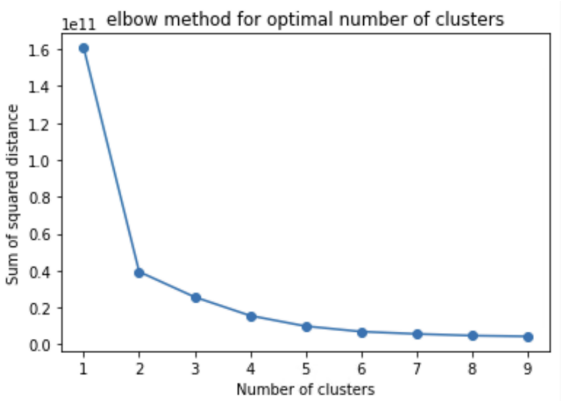


Figure 7: Elbow Plot

Dendrograms in Hierarchical Clustering The dendrograms, depicted in figure 8, served as a visual of the clustering outcomes. When looking at the dendrogram, the amount of 2 clusters seemed optimal. Cutting the dendrogram at a threshold of 3000 gives us two clusters and five outlier clusters.

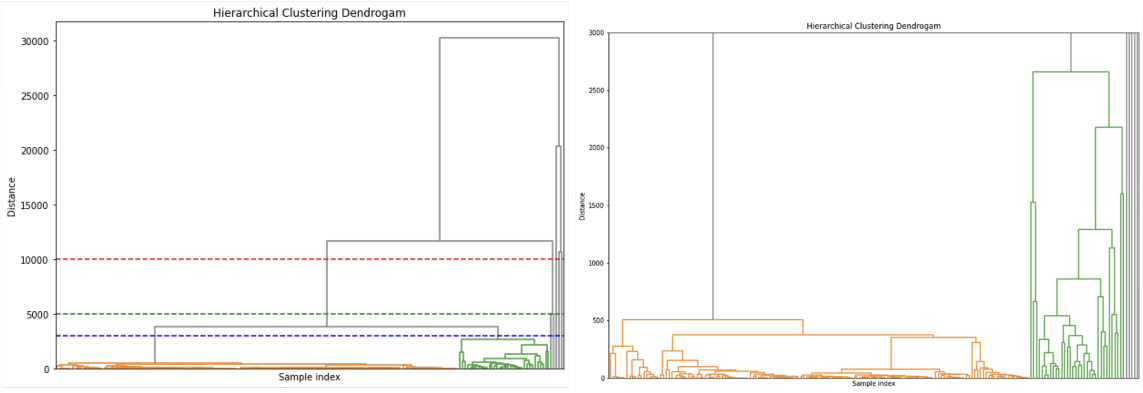


Figure 8: Dendrograms

5.2.2 Cluster 1 Analysis

Cluster 1 contains 150 events, which accounts for 79% of the total 189 events. Each event is visualized in separate subplots to analyze their vitals. Figure 9 shows a few of these graphs, showing relatively stable events. Especially looking at the heart rate, there are few fluctuations. After analyzing all the plots in Cluster 1, it becomes clear that the majority of events show stable patterns, while just a small number display more fluctuations of the vitals. To summarize, cluster 1 consists of incorrectly assigned events, as they show stability and lack significant event or artifact presence.

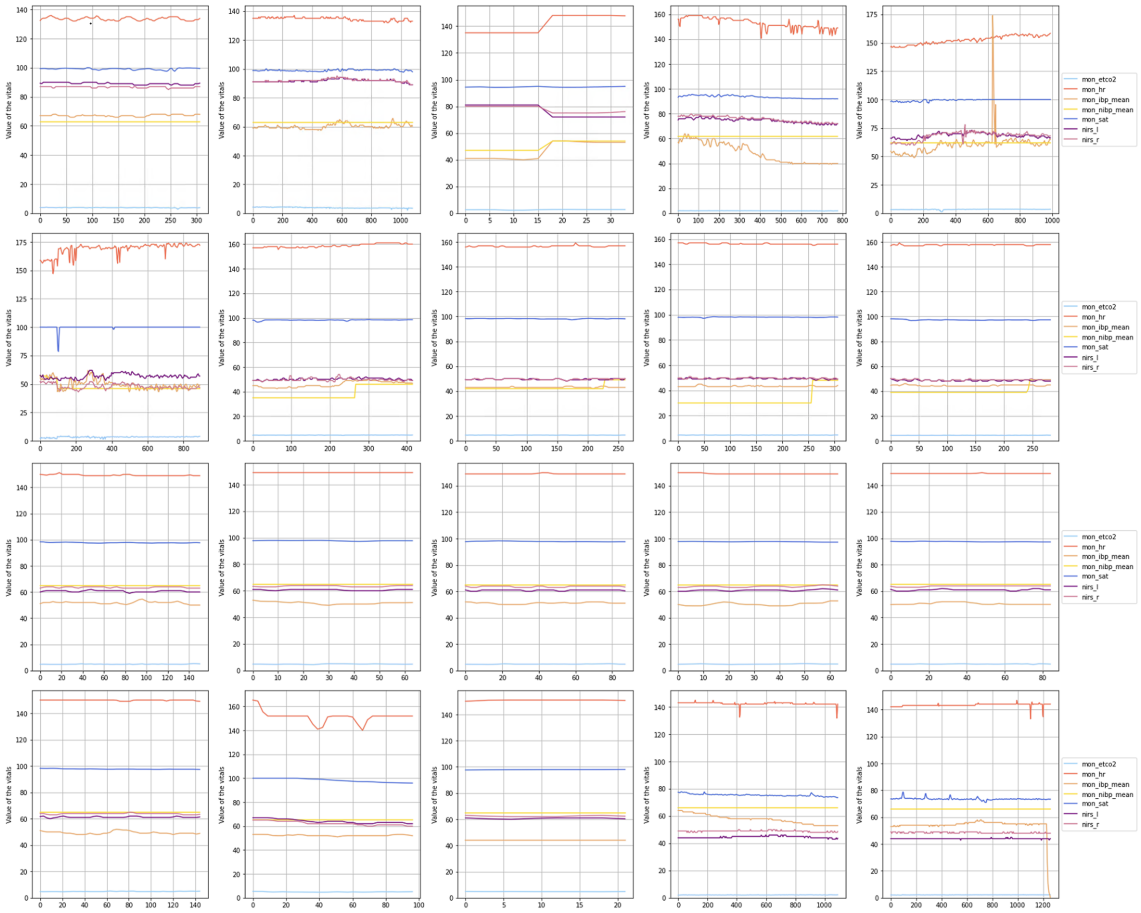


Figure 9: Cluster 1

5.2.3 Cluster 2 Analysis

Cluster 2 contains 34 events out of a total of 189. Figure 10 shows each event in subplots to help with the analysis of their vitals. Cluster 2 shows unstable patterns in contrast to Cluster 1, which includes a greater amount of events with a higher level of variability and fluctuations in the plots. The plots show sudden and fast fluctuations in the vital values. This suggest that these events could be real events or artifacts. Specifically, when one of the vitals has a rapid and significant change, it likely indicates an event. However, in order to accurately identify these situations, an clinical professional is needed to look at them for the right interpretation. Nevertheless, the patterns seen in Cluster 2 indicate that the GRU model is capable of extracting and classify events from the PICU data. Therefore, showing its potential usefulness in enhancing alert management systems.

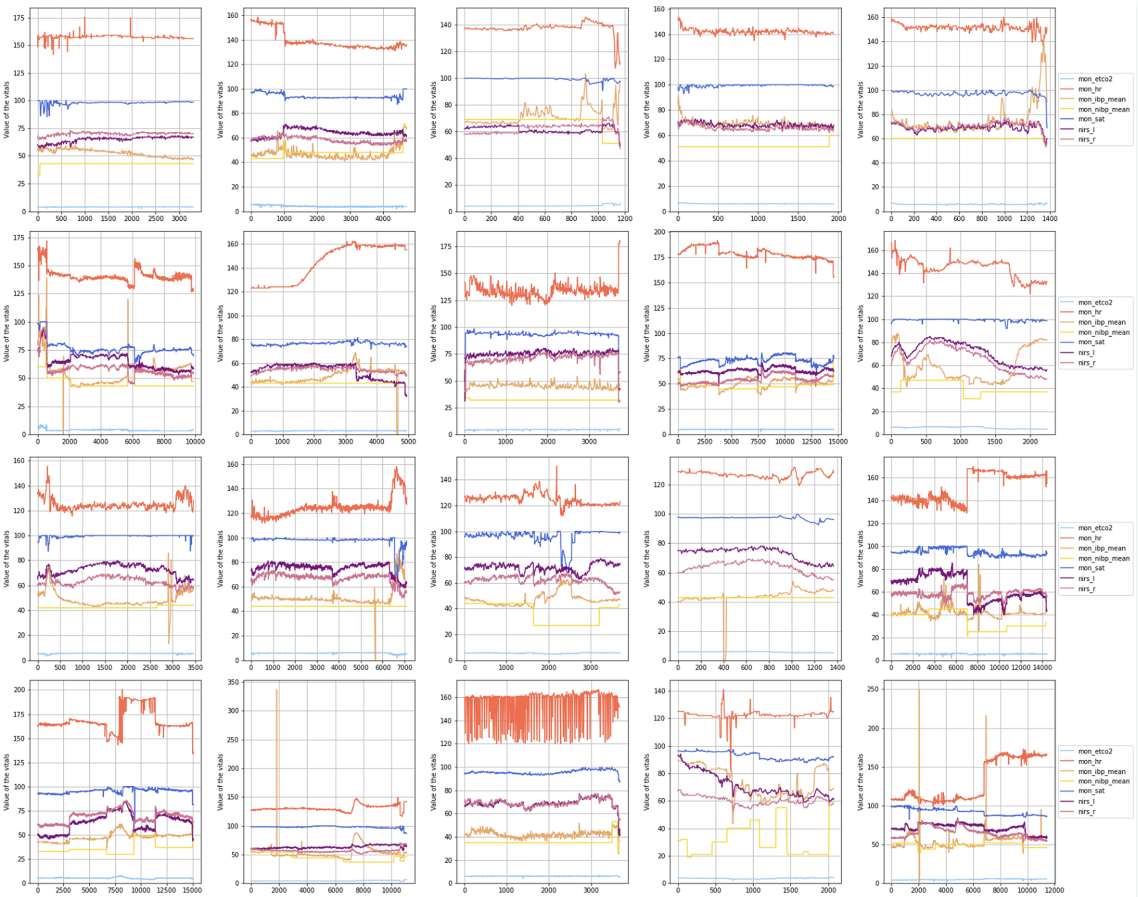
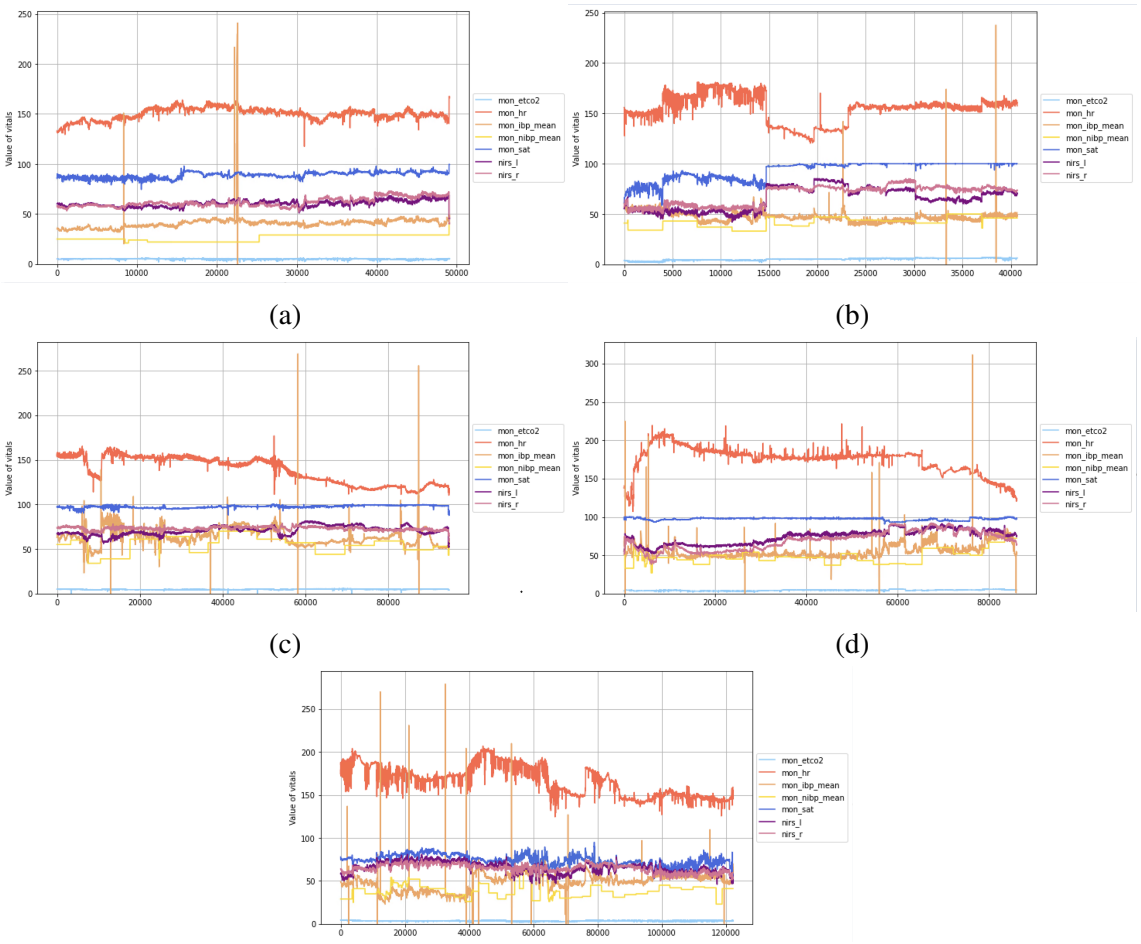


Figure 10: Cluster 2

5.2.4 Outlier Clusters

The outlier clusters represent events that do not fit into the main clusters regarding the dendrogram from the hierarchical agglomeration clustering. Figures 11 show examples of such outlier segments. These outliers can be important for identifying unusual events. When analyzing the visualizations of the outliers they may be events or artifacts. However they differ from cluster 1 and cluster 2, due to their long duration. This could indicate that these "outliers" show the long events when the patient is experience difficulties. However, an clinical professional is needed to look at them for the right interpretation.



(a) (b)

(c) (d)

(e)

Figure 11: Outliers

6 Discussion

This study investigated the use of Gated Recurrent Units (GRUs) to identify important clinical events in pediatric intensive care units (PICUs) to combat alarm fatigue. The study mainly explored the efficacy of GRU models in combination with a HAC model to detect event patterns and, therefore, aimed to enhance event detection models.

6.1 Summary of Results

The main results indicated that the GRU model had a high recall rate but faced challenges with precision. More specifically, The model shows high precision for the negative class, which are the non-events. It was difficult to predict the positive class, which are the events. Therefore, the accuracy of 84% is mainly due to the non-event class. However, the clustering provided additional insights by finding two clusters of events and multiple outliers. Cluster 1 consisted mainly of stable and, therefore, none event periods, while cluster 2 included more variate periods with high fluctuations. This indicates that cluster 2 consists of events or artifacts which are medically relevant. Overall, the clustering didn't distinguish different events but showed promising results in enhancing precision in event detection. Combining the GRU model with clustering allows us to filter out the event periods, as seen in cluster 2. These results show the potential of GRUs when used with hierarchical agglomerative clustering to improve alarm systems in PICUs. These models can effectively identify significant clinical events, which is a step toward prioritizing alarms that need immediate attention. This will improve patient safety and reduce the mental workload for healthcare providers. However, the high number of false positives suggests the need to improve the model. Besides, an expert is needed to confirm if the events in cluster 2 are accurate.

6.2 Limitations

This study gathered valuable insight into detecting events from PICU data. However, certain limitations were encountered that could have influenced the study results.

1. *Manual Labeling* A non-expert manually assigned labels to the events. This may have introduced biases and inconsistencies. While manual labeling was necessary to train the model, this strategy can influence the accuracy and generalizability of the outcomes.

2. *Clinical Validation* Clinical professionals must review the clusters and outliers identified by the model to confirm their significance. Without this expert validation, we cannot be sure about the actual interpretations of the clusters from this study.

3. *Virtual Machine Issues* Using a virtual machine (VM) without internet access to protect patient data caused several problems. Sometimes, the VM malfunctioned, and downloading necessary packages was impossible. The VM also had low RAM considering the size of the dataset, which made handling a large dataset difficult and took the research into the direction of complete case analysis, leading to less data. Computations often took a long time, sometimes up to half a day, which was a significant hurdle given the tight schedule of the research.

6.3 Recommendations

Future studies should focus on several areas to overcome the limitations and build upon this study's findings.

1. *Model Refinement* Directions for future research could include feature extraction or providing only one or two features as input to focus on, such as only looking at the heart rate. It would be interesting to see if the model could be improved by using a different architecture.

2. *Clinical Collaboration* As mentioned before, the labeling process was done by a non-expert. Using clinical professionals in both the labeling process and the interpretation of data could improve the outcome. Their help will ensure that the results are clinically relevant. therefore enhancing the model's precision and dependability.

3. *Extended Validation* The model could be better trained by adding more labeled data. Therefore, the model's ability to generalize and withstand variations would be improved, resulting in better prediction. Besides providing additional labeled data, the validation of the model would also be more trustworthy.

4. *Virtual Machine Improvements* The virtual machine setup can be improved to overcome its existing limitations. Future research that uses a VM should prioritize the amount of RAM and explore methods to overcome limitations on package downloads caused by internet restrictions. Also, using more powerful virtual machines could decrease computation times and improve efficiency. However, balancing the safety of patient data with easy access to a productive work environment can be difficult. However, the priority must always be the safety of patient data.

7 Conclusion

This study looked into how Gated Recurrent Units (GRUs) can help detect significant clinical events to reduce alarm fatigue in pediatric intensive care units (PICUs). Alarm fatigue is a serious problem in PICUs, where constant alarms can make healthcare providers less responsive. This leads to potentially missing critical events and putting patient safety at risk.

The GRU model showed a high recall rate, suggesting it can effectively identify critical events. This is especially crucial in PICUs, where timely detection of patient condition changes can be lifesaving. However, the model also had a high rate of false positives. While catching every potential event is important, too many false alarms can contribute to alarm fatigue, which defeats the purpose. By adding hierarchical agglomerative clustering (HAC), the model could provide two clusters, one with stable events and one with real events or artifacts. This combination with HAC showed very promising results. However, an expert should analyze the clusters to make assumptions about whether the events are clinically relevant.

Using real patient data and advanced machine-learning techniques were used in this research. The real patient data set of complex clinical situations makes our findings more relevant for future use. However, working with this data set brought challenges since the data wasn't labeled. Therefore, manual labeling of events had to be done. This process was done by a non-expert and, therefore, could introduce biases and inconsistencies. This can affect the model's accuracy. We would need clinical professionals to validate the clusters and outliers we found. Without their input, we can't be sure about the significance of these patterns.

The virtual machine (VM), which was set up to protect patient data, faced technical challenges. This setup caused problems with downloading necessary packages and long computation times due to limited RAM. Especially with such a large dataset, it slowed down the research process. A stronger VM would be recommended for future research.

Overall, this study shows a significant potential of GRUs in combination with hierarchical agglomerative clustering (HAC) to detect events. This has the potential to address alarm fatigue and improve patient safety and care quality in PICUs. The study's results provide a promising start for future research to create more effective and sustainable solutions for alarm management in critical care settings.

References

[1] A. Abernethy, L. Adams, M. Barrett, C. Bechtel, P. Brennan, A. Butte, J. Faulkner, E. Fontaine, S. Friedhoff, J. Halamka, M. Howell, K. Johnson, P. Lee, P. Long, D. McGraw, R. Miller, J. Perlin, D. Rucker, L. Sandy, L. Savage, L. Stump, P. Tang, E. Topol, R. Tuckson, and K. Valdes. The promise of digital health: Then, now, and the future. *NAM Perspectives*, 2022. doi: 10.31478/202206e. URL `https://doi.org/10.31478/202206e`. Discussion Paper, National Academy of Medicine, Washington, DC.

[2] S. Sendelbach and M. Funk. Alarm fatigue: A patient safety concern. *AACN Advanced Critical Care*, 24(4):378–388, 2013. doi: 10.1097/NCI.0b013e3182a903f9.

[3] K. Lewandowska, M. Weisbrot, A. Cieloszyk, W. Mędrzycka-Dąbrowska, S. Krupa, and D. Ozga. Impact of alarm fatigue on the work of nurses in an intensive care environment—a systematic review. *International Journal of Environmental Research and Public Health*, 17(22):8409, 2020. doi: 10.3390/ijerph17228409.

[4] K. J. Ruskin and D. Hueske-Kraus. Alarm fatigue: Impacts on patient safety. *Current Opinion in Anaesthesiology*, 28:685–690, 2015.

[5] M. Cvach. Monitor alarm fatigue: An integrative review. *Biomedical Instrumentation & Technology*, 46(4):268–277, 2012. doi: 10.2345/0899-8205-46.4.268.

[6] R. Venkateswaran, K. Kreitzman, M. Fraai, E. Brosseau, L. Preston, and B. Scirica. Abstract 190: A systematic alarm management pilot to reduce alarm fatigue at a tertiary care center. *Circulation: Cardiovascular Quality and Outcomes*, 12, 2019. doi: 10.1161/hcq.12.suppl_1.190.

[7] O. M. Cho, H. Kim, Y. W. Lee, and I. Cho. Clinical alarms in intensive care units: Perceived obstacles of alarm management and alarm fatigue in nurses. *Healthcare Informatics Research*, 22(1):46–53, 2016. doi: 10.4258/hir.2016.22.1.46.

[8] J. Chromik, S. A. I. Klopfenstein, B. Pfitzner, Z. C. Sinno, B. Arnrich, F. Balzer, and A. S. Poncette. Computational approaches to alleviate alarm fatigue in intensive care medicine: A systematic literature review. *Frontiers in Digital Health*, 4:843747, 2022. doi: 10.3389/fdgth.2022.843747.

[9] K. Merkelbach, S. Schaper, C. Diedrich, et al. Novel architecture for gated recurrent unit autoencoder trained on time series from electronic health records enables detection of icu patient subgroups. *Scientific Reports*, 13(1):4053, 2023. doi: 10.1038/s41598-023-30986-1. URL `https://doi.org/10.1038/s41598-023-30986-1`.

[10] Mustafa Hussain, James Dewey, and Nadir Weibel. Reducing alarm fatigue: exploring decision structures, risks, and design. *EAI Endorsed Transactions on Pervasive Health and Technology*, 3(10), 2017.

[11] CM Pater, TK Sosa, J Boyer, et al. Time series evaluation of improvement interventions to reduce alarm notifications in a paediatric hospital. *BMJ Quality & Safety*, 29:717–726, 2020. doi: 10.1136/bmjqs-2019-010184.

[12] X. Wang, Y. Gao, J. Lin, H. Rangwala, and R. Mittu. A machine learning approach to false alarm detection for critical arrhythmia alarms. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 202–207, 2015. doi: 10.1109/ICMLA.2015.48.

[13] S. L. Hyland, M. Faltys, M. Hüser, X. Lyu, T. Gumbsch, C. Esteban, C. Bock, M. Horn, M. Moor, B. Rieck, M. Zimmermann, D. A. Bodenham, K. M. Borgwardt, G. Rätsch, and T. M. Merz. Early prediction of circulatory failure in the intensive care unit using machine learning. *Nature Medicine*, 26:364–373, 2020. doi: 10.1038/s41591-020-0789-4.

[14] Hwin Dol Park, Youngwoong Han, and Jae Hun Choi. Medical time-series prediction with lstm-mdn-attn. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1359–1361, 2019. doi: 10.1109/ICTC46691.2019.8939761.

[15] Zachary C. Lipton, David C. Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks, 2017.

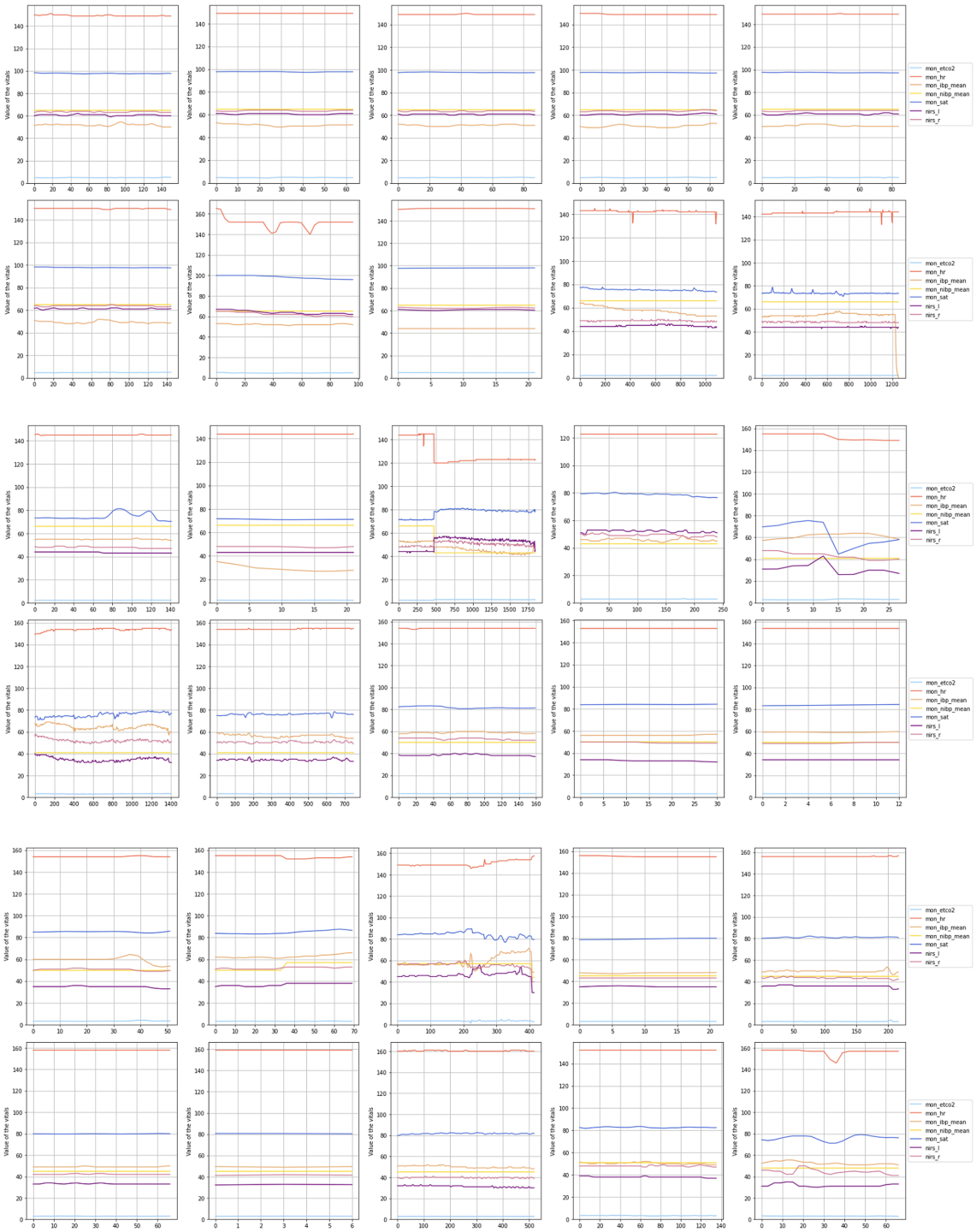
[16] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

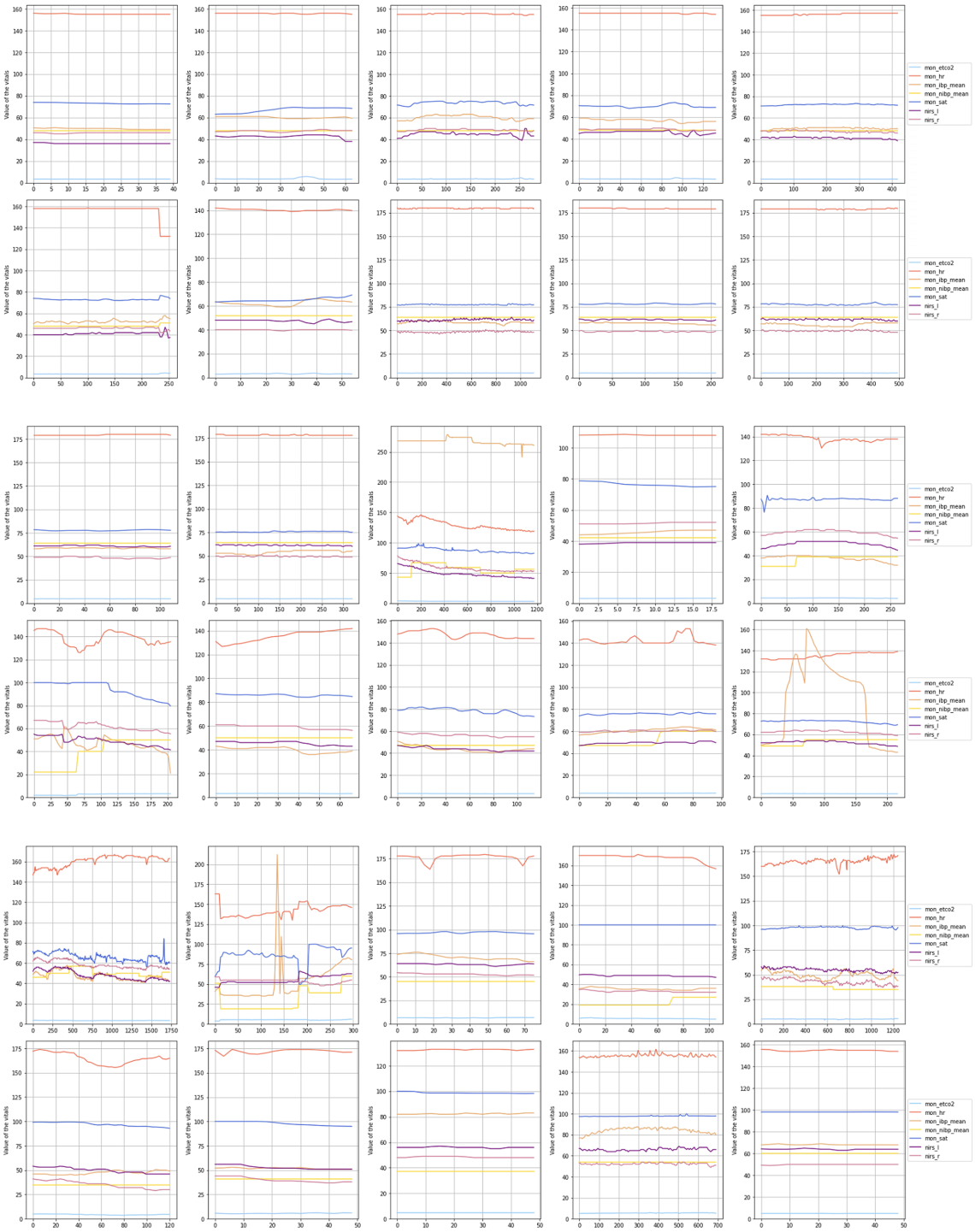
[17] WildML. Recurrent neural network tutorial, part 4 – implementing a gru/lstm rnn with python and theano – wildml. `https://wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn` 2015. Archived from the original on 2021-11-10. Retrieved May 18, 2016.

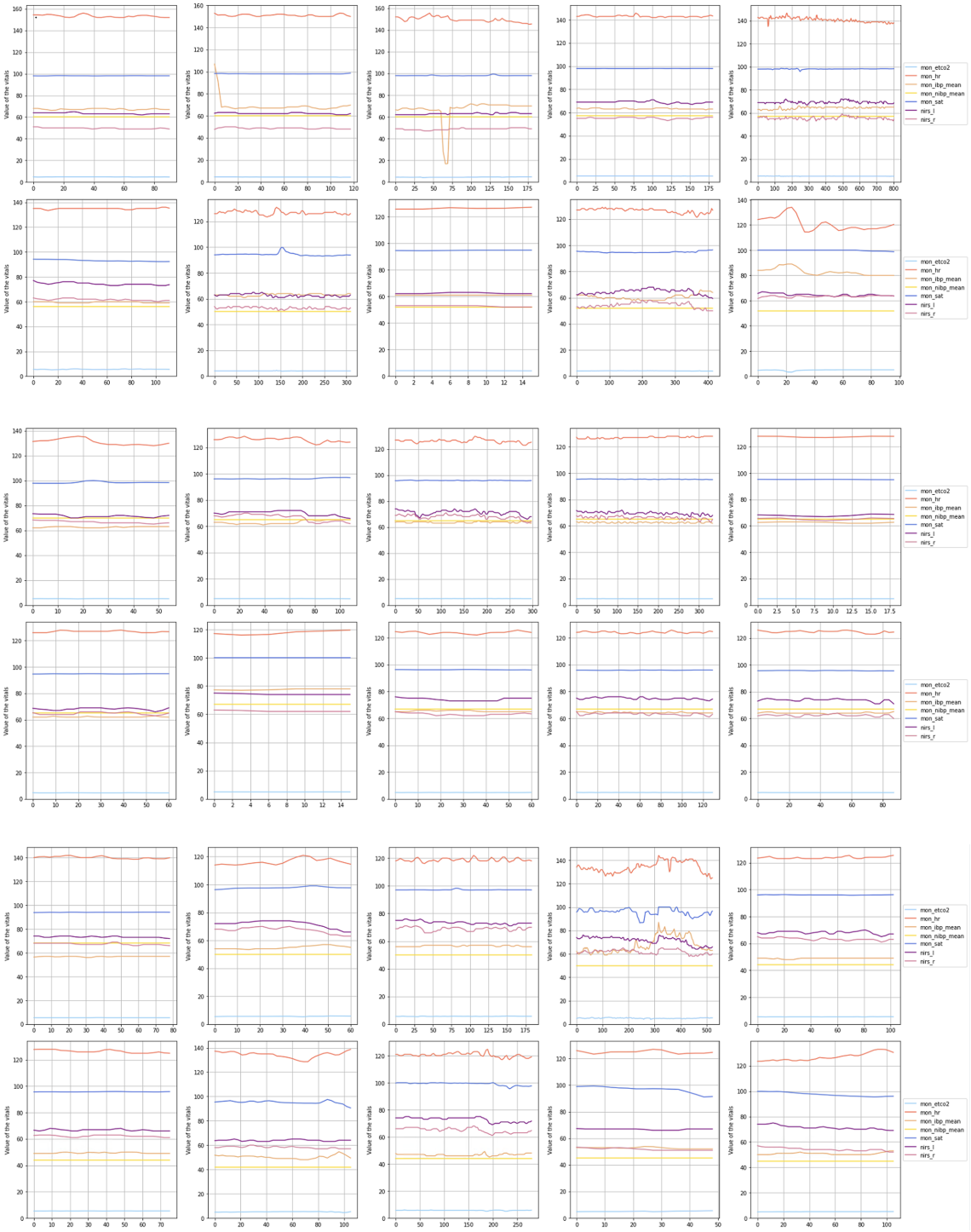
[18] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, pages 98–101, 2020. doi: 10.1109/IWECAI50956.2020.00027.

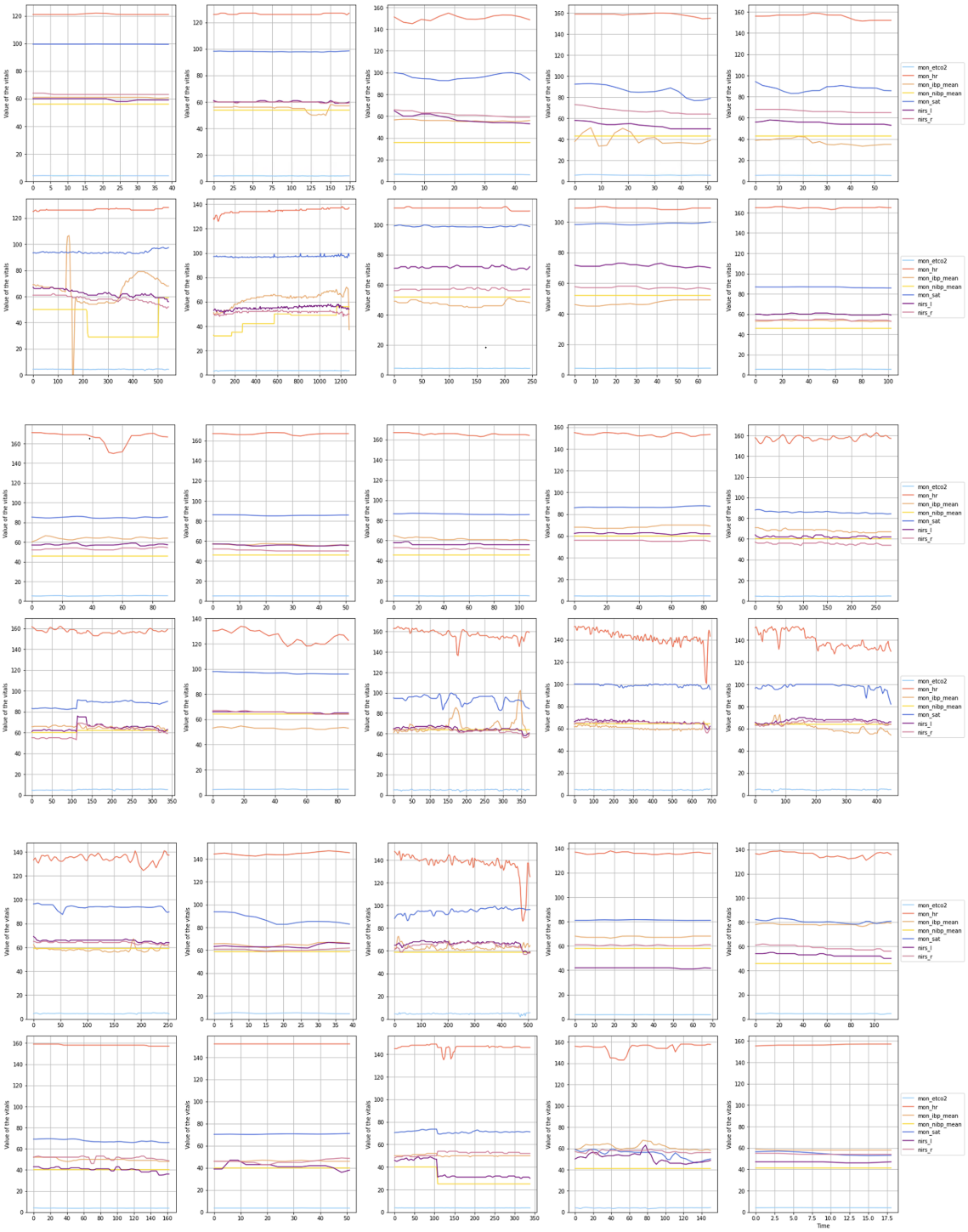
Appendix

A Cluster 1

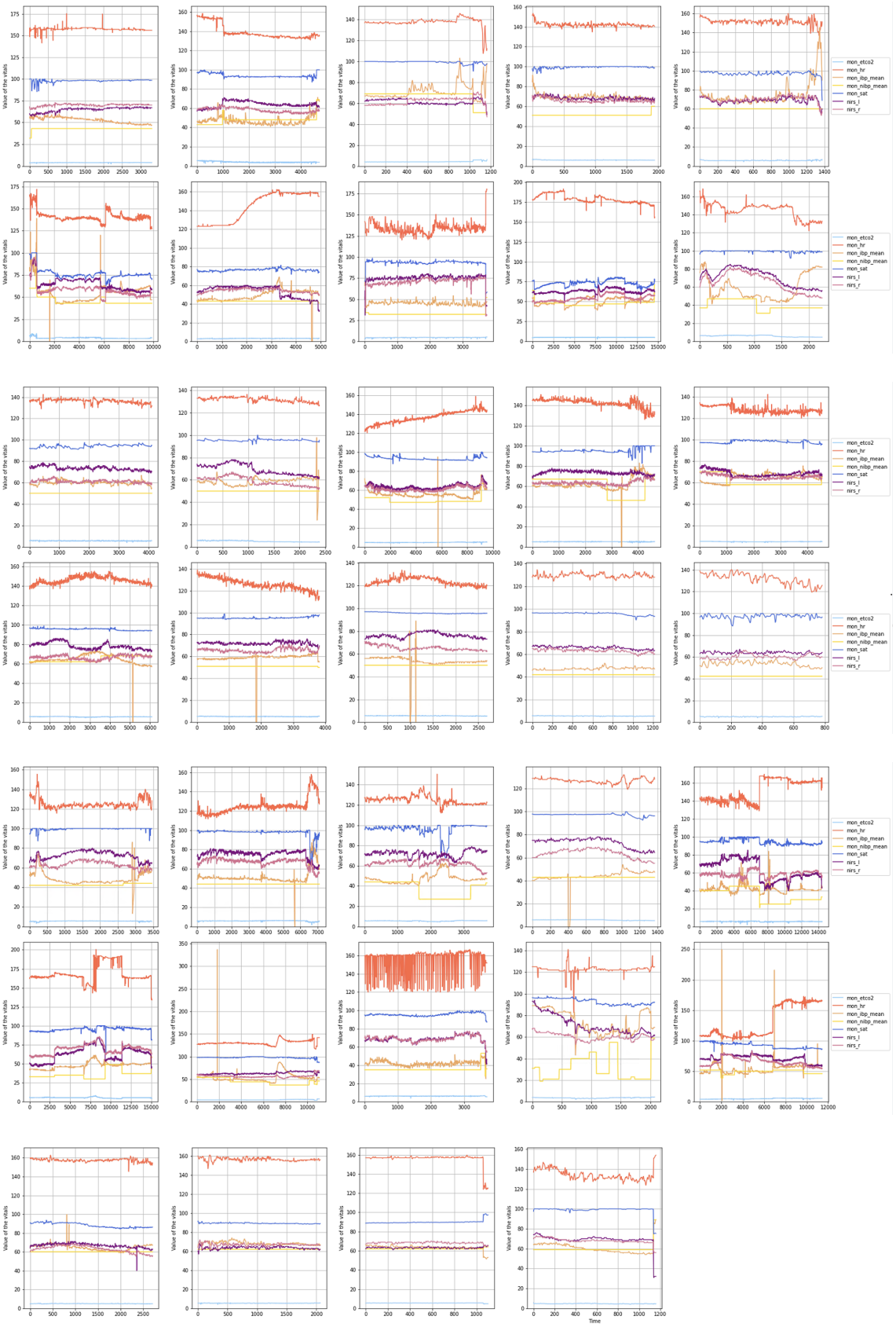

Value of the vitals
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r


Value of the vitals
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r

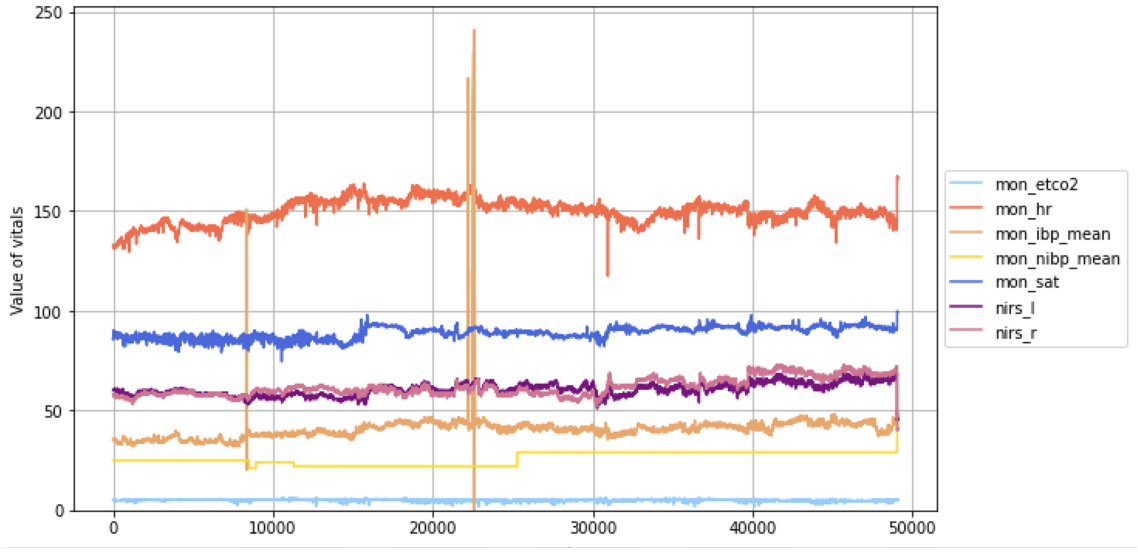

Value of the vitals
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r

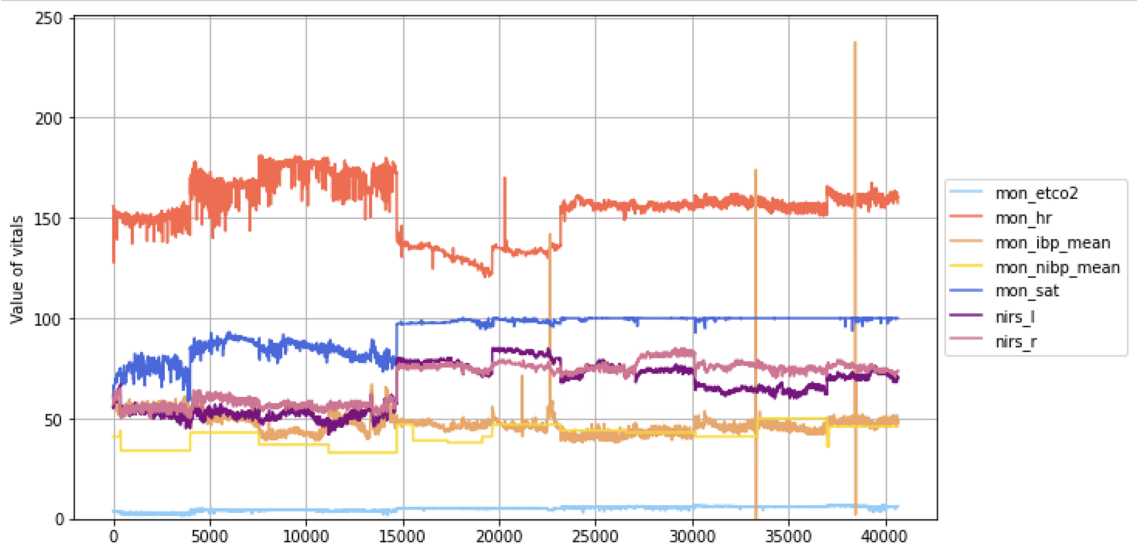

Value of the vitals
Time
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r

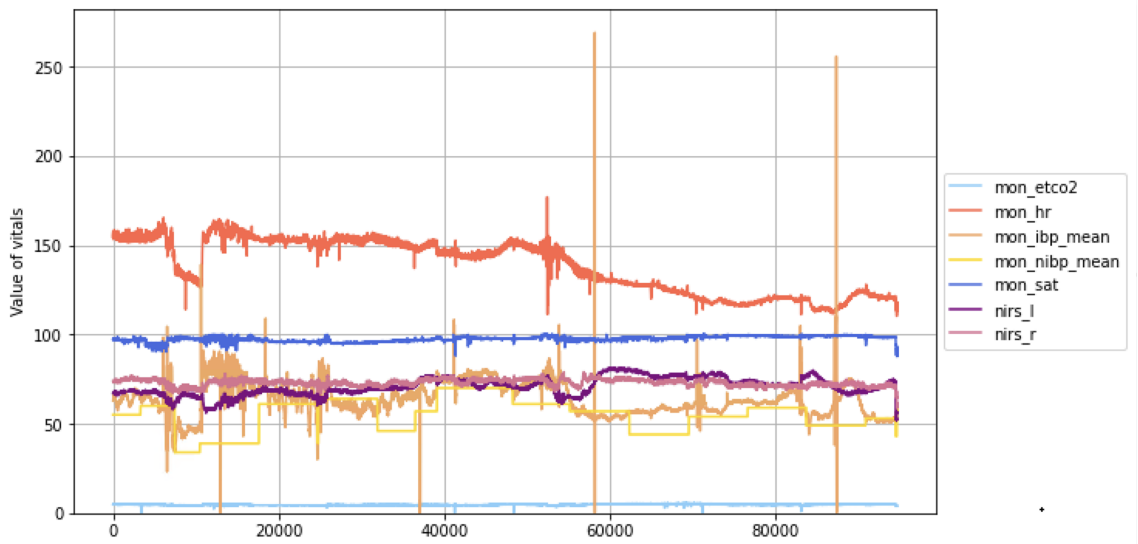
B Cluster 2

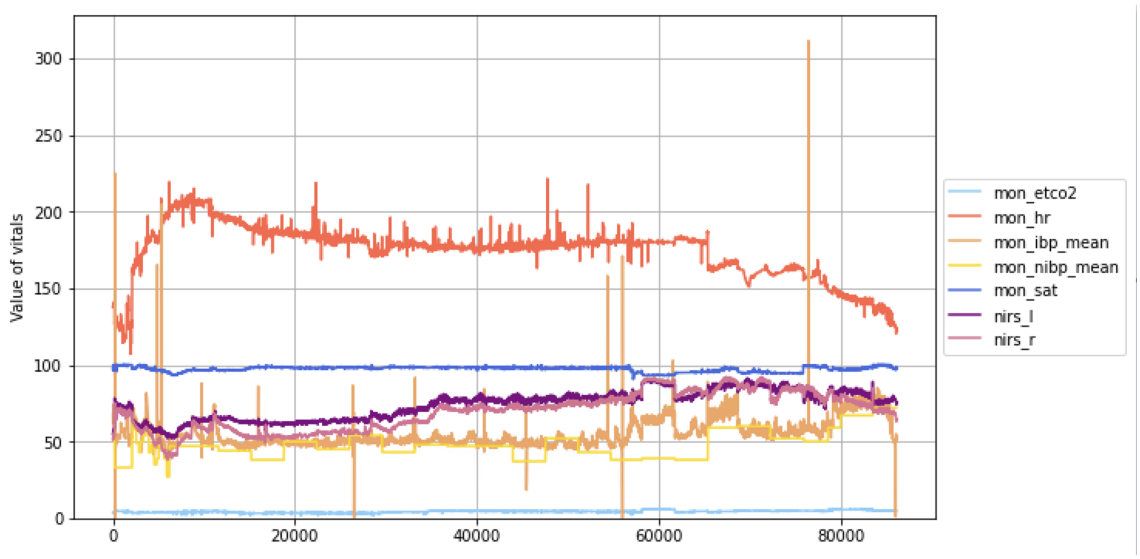

Value of the vitals
Time
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r

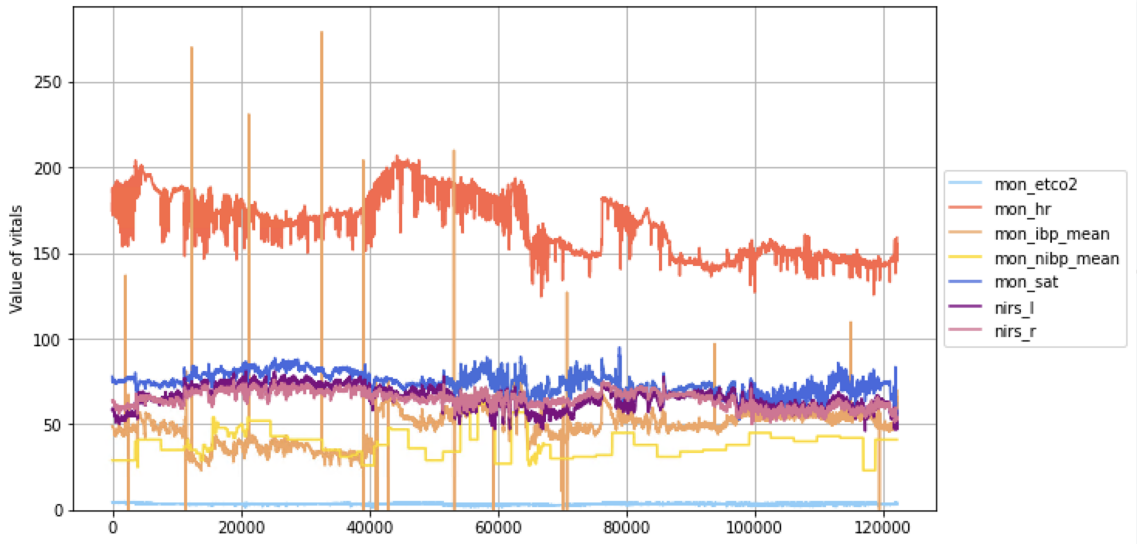
C Outliers


Value of vitals
0
50
100
150
200
250
0
10000
20000
30000
40000
50000
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r


Value of vitals
0
50
100
150
200
250
0
5000
10000
15000
20000
25000
30000
35000
40000
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r


Value of vitals
0
50
100
150
200
250
0
20000
40000
60000
80000
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r


Value of vitals
0
50
100
150
200
250
300
0
20000
40000
60000
80000
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r


Value of vitals
0
50
100
150
200
250
0
20000
40000
60000
80000
100000
120000
mon_etco2
mon_hr
mon_ibp_mean
mon_nibp_mean
mon_sat
nirs_l
nirs_r