ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CENTER FOR PROJECT-BASED LEARNING

Department of Information Technology and Electrical Engineering

Spring Semester 2023

Real-Time Prediction of Sepsis Onset with Machine Learning Models

Bachelor Thesis

Andrea Ghirlanda
aghirlanda@student.ethz.ch

June 2023

Supervisors: Dr. Kanika Dheman, kanika.dheman@pbl.ee.ethz.ch
Mr. Marco Giordano, marco.giordano@pbl.ee.ethz.ch

Professor: Prof. Dr. Sebastian Kozerke, kozerke@biomed.ee.ethz.ch

Acknowledgements

I would like to thank Marco Giordano, Dr. Kanika Dheman, and the Center for Project-Based Learning for the opportunity of doing this thesis.

Abstract

Sepsis is a life-threatening condition that affects 49 million people every year and is responsible for 11 million annual deaths. Early recognition of sepsis could lead to an increase in patient survival chances but detecting sepsis, especially in its early stages continues to be a medical challenge. In recent times, thanks to the increased availability of digital health data, an extensive literature has formed about the application of machine learning algorithms to detect sepsis. Having models that run on edge devices is of utmost importance for the preventive treatment of the disease.
In this thesis, approaches to creating a model that would fit in an MCU are explored. With an initial sliding window approach to the problem, the impact of padding on the ability of ML models to learn meaningful information is explored. Following this, an offline setting is used, and finally, various techniques to reduce model size with minimal impact on performance are investigated. The end result was a well-performing model with AUROC > 0.7 and AUPRC > 0.45 which had a weight of only approximately 120KB and a maximum activation size of 3.5KB.

Declaration of Originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Andrea Ghirlanda,
Zurich, June 2023

Contents

List of Figures

List of Tables

List of Acronyms

AUPRC . . . . . Area Under the Precision-Recall Curve
AUROC . . . . . Area Under the Receiver Operating Curve

BCE . . . . . Binary Cross-Entropy

CNUH . . . . . Chonnam National University Hospital

DBP . . . . . Diastolic Blood Pressure

GRU . . . . . Gated Recurrent Unit

HiRID . . . . . High time-Resolution ICU Dataset
HR . . . . . Heart Rate

ICU . . . . . Intensive Care Unit

LSTM . . . . . Long-Short-Term Memory

MCU . . . . . Microcontroller Unit
ML . . . . . Machine Learning

NN . . . . . Neural Network

qSOFA . . . . . quick Sequential Organ Failure Assessment

ReLU . . . . . Rectifier Linear Unit

List of Acronyms

RNN . . . . . .Recurrent Neural Network

RR . . . . . .Respiratory Rate

SBP . . . . . .Systolic Blood Pressure

SI . . . . . .Suspicion of Infection

SIRS . . . . . .Systemic Inflammatory Response Syndrome

SOFA . . . . . .Sequential Organ Failure Assessment

SPO2 . . . . . .Oxygen Saturation

std . . . . . .Standard Deviation

TCN . . . . . .Temporal Convolutional Network

TF . . . . . .Tensorflow

TFLITE . . . . .Tensorflow Lite

UV . . . . . .University of Virginia

Chapter 1

Introduction

1.1 Motivation

Sepsis is a deadly syndrome that consists of a dysregulated immune response to infection that leads to organ dysfunction[1]. Each year 49 million people suffer from sepsis of which 11 million end up dying. This makes sepsis one of the leading causes of death globally accounting for 20% of worldwide deaths every year[2].

One of the main challenges in treating sepsis is its detection. Detecting Sepsis, especially in its early stages is difficult since for now, due to its systemic nature, no set of biomarkers is universally recognized for diagnosing and treating sepsis. For this reason, over the years many scores have been developed to identify sepsis: The first definition of Sepsis (Sepsis 1) was "the systemic response to infection, manifested by two or more of the Systemic Inflammatory Response Syndrome (SIRS) criteria as a result of infection". Later on, Sepsis had been redefined (Sepsis 2) where a lot of new criteria were added to account for the SIRS score. The last update to the sepsis definition came in 2016 when SIRS was substituted by the Sequential Organ Failure Assessment (SOFA) and the quick SOFA (qSOFA) scores [3].

However, these scores can only identify Sepsis when the disease has already reached a critical level. Therefore further methods are required to detect sepsis onset early enough so that it can be prevented. Machine Learning (ML), and in particular Neural Networks (NN) have already been extensively used to analyze time series and are showing promising results in early sepsis onset detection.
However, the open-source datasets available, such as "MIMIC III"[4], usually have a granularity of 1 sample per hour which does not allow for the identification of quick changes in biomarkers.

Additionally, previous research is based on a lot of variables[5], many of which, such as laboratory data, take a lot of time to be collected and analyzed, and cannot be acquired in many settings such as hospitals in underdeveloped countries and especially in home settings. It is therefore important to have wearable devices that can gather data from patients inside and outside of the hospital, and can give a warning in the case of incoming onset of sepsis.
For this reason, well-performing lightweight machine learning models that can fit in such wearable devices are necessary.

1.2 Objective

The general goal of the thesis was to find a lightweight model that uses only a few channels of data with a relatively short data window so that it would fit in an MCU for on-device inference. The low number of channels is needed in order for the model to require only data that a wearable device could collect and to reduce the model's size.

1.3 Research Questions

The first research question that this thesis aimed to answer was how the performance of an already implemented model would change by switching from an offline setting to an online setting using a high-granularity dataset.

Following this, the impact of onset matching and padding on the performance of models in an online setting was investigated.
Then, returning to an offline setting and doing a horizon evaluation, the influence of padding was explored further.

Finally, it was explored how various techniques, used to decrease the number of parameters in the model, such as quantization, max pooling, and pruning, would affect the performance of the model. Here also the impact of time resolution on the model performance during inference by training and testing it with a 60-minute resolution dataset was assessed.

1.4 Outline

After thorough literature research on the topic of early sepsis detection, the model which was the best fit for the goals of the thesis was evaluated, opting for a Temporal Convolutional Network (TCN) architecture.

TCN was chosen since it was already shown in the literature to be performing well [6, 7] but differently from other architectures like the transformer, it wasn't as heavy in terms of parameters, making it a good choice for a later MCU implementation.
After choosing the general architecture, an online setting was implemented where the model was fed all of the shifted windows from each patient sequentially before switching patients.

Later analysis on the effect of padding on the model's performance led back to an offline setting with a 4-hour data window on which a horizon evaluation with prediction times ranging from 1 hour up to 4 hours was done. Seeing that the model could perform well in this setting the next stage was preparing the model for inference on the MCU.
The first step was writing the code for the TensorFlow model, training, and testing it together with the data pipeline.
After that, the model quantization and the testing for the quantized model were implemented. This was followed by precise calculations of the model size. Subsequently, further ways to decrease the number of parameters of the model were implemented. Some optimizations worked well, such as max pooling, while others, such as pruning, were not included in the final model due to poor performance. Finally, the model was deployed on an MCU.

Chapter 2

Related Work

Paper	Dataset	Features	Model	Data Window	AUROC	AUPRC
[8] - 2023	MIMIC-III,Challenge[9]	7Vitals,21 Lab tests,4 demogr.	Semi-Supervised model	6h	0.76	-
[10] - 2022	MIMIC-III,Challenge	Vitals, Lab tests,demogr.,medical interventions	XGBoost	-	0.8257	-
[11] - 2023	CNUH,UV	7 Vitals, 24 lab tests	DGAT	8h-24h	0.937	0.865
[6] - 2022	Challenge	8Vitals,26Lab tests, 6demogr.	TCN	-	0.892	0.527
[7] - 2019	MIMIC-III	44Vitals,Lab	TCN	<48h	0.87	0.42
[12] - 2016	MIMIC-III	6Vitals, 1demogr.,GCS	InSight	-	0.74	0.28

Table 2.1: Summary of Relevant Publications

Table 2.1 summarizes some important publications and gives an overview of some relevant studies.
All of the publications mentioned in the table use the latest definition of sepsis (Sepsis-3) to set the true labels.

In this chapter, the literature surrounding sepsis early onset detection is explored in the different following sections.

2.1 Setting

Most of the studies published have been made by training and evaluating a model in an offline setting. Horizon evaluation has often been done. Horizon Evaluation gives useful information on how early sepsis is recognized by training and testing the model using different prediction times and looking at the change in performance over the various prediction times.

There are also a few studies that have been done in an online setting, where the data

window gets shifted by a set time interval after each prediction. This mimics a real-world scenario where new data is constantly gathered and fed to such a model.[5].

2.2 Inclusion Criteria

Furthermore, there are various inclusion criteria being used in different studies.
Some of the most common ones are:

- Infection at the time of admission [13]
- The patient must be old enough (e.g. >18y.o. [13, 14], >14y.o [15]
- Not too much missing data[13, 14, 15]
- Long enough ICU stay (e.g >6h [15]
- Onset during ICU stay [15]
- Onset not too early or too late (e.g before 4h of ICU stay and after 168h of ICU stay [15]; before 7h of ICU stay and after 2000h of ICU stay [14])

These criteria are often chosen in order to ensure that enough data is available from each individual patient

2.3 Models

Previous studies have mainly been carried out using Recurrent Neural Networks (RNN) with Gated Recurrent Unit (GRU), Temporal Convolutional Networks (TCN), Transformer, and Boosted Tree models [5].
Other models such as semi-supervised models have also been used [8].
Additional models used are also variations of the models listed above. For example a study from 2019 [7] uses a Gaussian Process Temporal Convolutional Network, which, as the name suggests, applies Gaussian Processes to a Temporal Convolutional Network.

Work has also been done in designing a good loss function over which to optimize. An article from 2023 [16], for example, devised an objective function called "Temporal Label Smoothing" (TLS) which could allow for better performance in an online setting by having a soft label that monotonically increases as the data window approaches the onset time, thereby increasing the penalty for wrongly classified positive patients as the time-to-onset is reduced.

The Smoothing function used in the paper is the following:

$$q(t) = \begin{cases} 0 & \text{for t} \leq t_e - h_{max} \\ e^{-\gamma(t_e - t - d) + A} & \text{for t} \in (t_e - h_{max}, t_e - h_{min}) \\ 1 & \text{for t} \geq t_e - h_{min} \end{cases} \tag{2.1}$$

where h_{min} and h_{max} define the range over which smoothing is applied, γ controls the smoothing strength, the parameters A,d are defined in order to make the function continuous and t_e is the onset time. Fig. 2.1 displays the above mentioned function graphically.

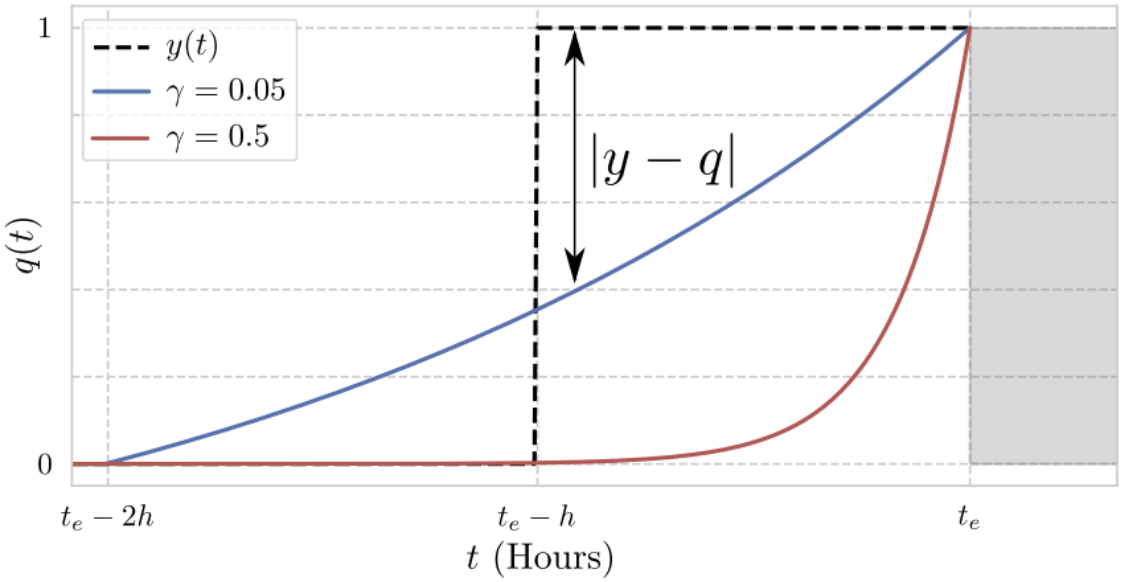

Figure 2.1: Temporal Label Smoothing Function

2.4 Data

Most studies have used many different biomarkers, including lab values, which can be time-consuming to gather and not available in many scenarios[5, 8, 7]. This is also shown in the summary table 2.1.
The most widely used dataset is the MIMIC-III dataset, which can also be seen in the table above.

MIMIC-III is a dataset containing health-related data of more than 40000 patients from

the ICU of Beth Israel Deaconess Medical Center in the time period spanning from 2001 and 2012 [4]. This dataset contains demographics information, vital signs measurements, along with laboratory test results, mortality and notes from the doctors. MIMIC-III's frequency is around 1 datapoint each hour.

Some studies also use private datasets. A publication from 2023 [11], for example uses the Chonnam National University Hospital (CNUH) dataset.

2.5 Metrics

When working with highly imbalanced datasets, accuracy is not a good metric since it could be high even for a classifier that only predicts one label. For this reason the Receiver Operating Curve and the Precision Recall Curve are used when the dataset is imbalanced. The Receiver Operating Curve displays the true positive rate (TPR) against the false positive rate (FPR) with varying thresholds in the range $[0, 1]$ where:

$TPR = \frac{TP}{TP+FN}$ and $FPR = \frac{FP}{FP+TN}$

The Precision Recall Curve, on the other hand, displays the Precision against the Recall, with varying thresholds in the range $[0, 1]$ where:

$Precision = \frac{TP}{TP+FP}$ and $Recall = TPR$

In order to extract a scalar value from those curves, thereby having a number that indicates the performance of a model using those metrics, the integral over the threshold is taken. The end result of this are the Area Under the Receiver Operating Curve (AUROC) and the Area Under the Precision-Recall Curve (AUPRC).

The metric that has most been reported is the Area Under the Receiver Operating Curve, which as highlighted by a review from Moor et. al.[5] could still be susceptible to class imbalance and should be reported together with the Area Under the Precision-Recall Curve for a good measure of the actual model performance. Unfortunately, many studies have not reported the AUPRC.

2.6 Comparison

Together with the above-mentioned metrics problem that the AUPRC is not often reported, there are other problems that make it difficult to compare the performance of models across studies [5].
First of all, the setting chosen (online or offline) greatly influences the performance of the models which makes it difficult to compare publications that use two different settings.

Moreover, the method used for onset matching 3.9.4 can also be a decisive factor in determining the performance of a model, especially on online settings, since this choice could greatly vary the amount of data taken from each individual negative patient.
Additionally, although all the publications in the table above 2.1 use the latest definition of sepsis (sepsis 3), other publications sometimes use different definitions to assign the labels.
Finally many publications don't disclose important details of their approach such as the above mentioned onset matching modality and whether they are training and evaluating their model in an offline or online setting. Some publications also don't disclose the prediction time and the window size which are also factors of great importance.

Overall, although vast literature is present in the field, comparing the results of different publications is usually very difficult and the problem of sepsis early detection still remains a challenge.

Chapter 3

Methods

3.1 Dataset

The dataset that was used throughout the thesis is HiRID (High Time-Resolution ICU Dataset). This dataset contains around 34000 patients from the ICU in the Bern University Hospital which was the result of a cooperation between the hospital and ETH Zürich.
HiRID contains deidentified demographic information together with 681 variables split among physiological variables, diagnostic test results, and treatment parameters, all with a 2-minutes time resolution[17].
As will be explained in section 3.3, one of the criteria the filtering was based upon, was the Admission-to-Sepsis time, which is the time period that it took for the patient to develop sepsis after being admitted into the hospital. This criterion was set for each patient to be >8h. By looking at figure 3.1, it can therefore be highlighted that around $\frac{2}{3}$ of the positive patients are discarded since most of them develop sepsis early on in their stay. Therefore only 607 out of the total 1851 positive patients are used.
After filtering according to 3.3, the total number of patients is 6758. This means that the class imbalance is $\frac{6758-607}{607} = 10.13$ negative patients for each positive sample.

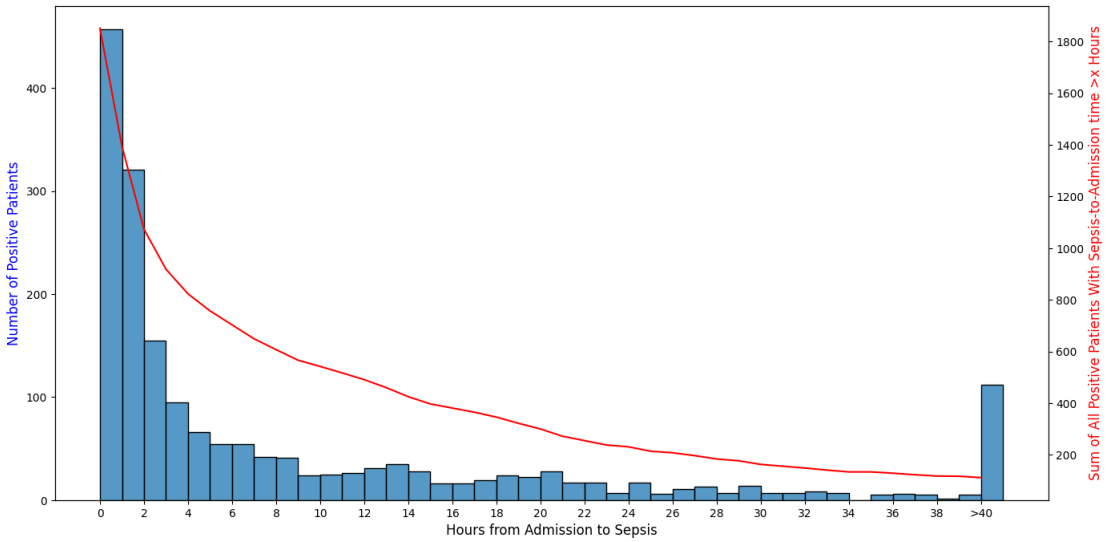

Figure 3.1: Admission-to-Sepsis Distribution of Positive Patients in HiRID Dataset

3.2 Assigning the Labels

The labels were assigned by checking the first time a suspicion of infection (SI) occurs. An SI is defined as a rapid increase of the SOFA score by 2 points.

Checking for an SI therefore simply consists of taking the difference of the SOFA score at time t and the SOFA score at time $t-1$. SOFA scores are assigned according to the table in Fig. 3.2, where the values of various organ systems are represented by a specific score number in the range $[0, 4]$.

If no SI is found, then the patient is labeled as negative. Otherwise, if the patient is positive, it is checked if any antibiotics were given in the 48 hours before and after the SI in order to assess whether the organ deterioration is caused by a bacterial infection. If antibiotics were administered in that time interval then the patient is considered positive and an onset time corresponding to the SI is returned, otherwise, the patient is deemed negative.

SOFA SCORE TABLE

Organ system	SOFA score				
	0	1	2	3	4
Respiratory, PO_2/FiO_2, mmHg (kPa)	≥400 (53.3)	<400 (53.3)	<300 (40)	<200 (26.7) with respiratory support	<100 (13.3) with respiratory
Coagulation, Platelets, $\times 10^3/mm^3$	≥150	<150	<100	<50	<20
Liver, Bilirubin, mg/dL	<1.2	1.2–1.9	2.0–5.9	6.0–11.9	>12.0
Cardiovascular	MAP ≥70 mmHg	MAP <70 mmHg	Dopamine <5 or dobutamine (any dose)[b]	Dopamine 5.1–15 or epinephrine ≤0.1 or norepinephrine ≤0.1[b]	Dopamine >15 or epinephrine >0.1 or norepinephrine >0.1[b]
Central nervous system, Glasgow Coma Scale	15	13–14	10–12	6–9	<6
Renal, Creatinine, mg/dL. Urine output, mL/d	<1.2	1.2–1.9	2.0–3.4	3.5–4.9 <500	>5.0 <200

Figure 3.2: SOFA Score Table [18]

3.3 Inclusion Criteria

For this project, three different inclusion criteria were defined in order to filter the patients available in the HiRID dataset:

- Length of stay $>$ 24h
- Admission-to-onset time $>$ 8h
- No antibiotics given in the first 8h before admission

The first two criteria were chosen in order to have enough data from the patient to feed to the model while the last criterion was needed in order to be able to give an accurate true label.

3.4 Train/Test Split and Sampling

For all experiments, a 80/20 train/test split was used as it is a standard way to split data. The training set was undersampled in order to have a 50/50 split of negative-to-positive patients which means that in each epoch the model sees a bit more than 1000 samples. With this method, this means that in every epoch, the model sees all of the positive patients and a subset of the negative patients in the interval of negative patients comprised in the following interval:

$[totalNrPositiveSamples * epoch, totalNrPositiveSamples * (epoch + 1)]$

3.5 Vital Signs

To train the models used in the experiments done in this thesis, the following 8 vital signs were used:

- systolic blood pressure (sbp)
- diastolic blood pressure (dbp)
- mean blood pressure (mbp) (mean between systolic and diastolic blood pressure)
- heart rate (HR)
- respiratory rate (RR)
- spO2
- core body temperature
- blood glucose

These vital signs were chosen due to the fact that they are easily acquirable with sensors on a wearable device. Additionally, these biomarkers provide a good overview of the general health status of a patient and they are among the most commonly used in healthcare.
Furthermore, as will be explained more in depth in section 3.12, for the final model blood glucose and mean blood pressure were discarded and the model was trained with the remaining 6 biomarkers.

3.6 Pre-Processing

In order to pre-process the data, first and foremost peaks were removed from all of the vital signs. This was done by looking at the ratio of the value at time point t to time point $t-1$. If the ratio is > 1.5, then the value at time point t is assigned the same value as the previous timepoint. Otherwise, no changes are made.

Each vital sign was then normalized using a min-max scaler that squeezes the values into the range [0, 1]. In the torch implementation, which corresponds to experiments 1 to 3, the scaler was newly fitted for each batch. The optimal way of scaling the data would be to fit the scaler using the whole dataset, in order to apply the same transformation to every sample in the dataset. However, fitting the scaler once for every batch does not impact the performance of the model significantly since, as shown in fig 4.6 it generalizes well even with smaller batch sizes.

3.6.1 Padding

The last point to mention is that when the patient does not have enough data, the window is padded with backward fill, which propagates the first available value backwards until the data window is completely filled. Fig. 3.3 shows the padding implementation on an individual channel (e.g. Heart Rate).

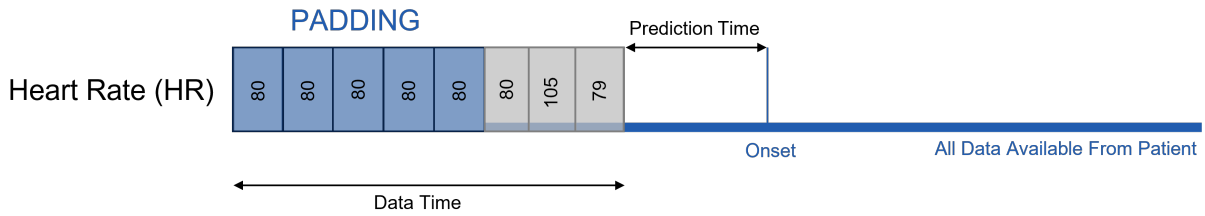

Figure 3.3: Visual Representation of Padding Implementation for a channel (Heart Rate)

3.7 Training

To train the network, Binary Cross-Entropy (BCE) Loss was chosen as it is the standard loss function for binary classification. Focal loss was also tried out but was ultimately not used.
For the optimizer, Adam was used as it is the default choice for training deep learning models. For the optimizer, a learning rate starting at 0.001 was chosen. Moreover, a scheduler was also used that divided the learning rate by 5 every 4 epochs. This was done in order to get closer and closer to the bottom of the local optimum found.

3.8 Model

Within this thesis, various models were initially considered, among some of which were Transformer, RNN, and TCN.
Transformer was discarded since it requires a lot of parameters by default which makes it particularly difficult to optimize for an implementation on an MCU.
RNN generally shows reduced performance compared to TCN while not providing particular benefits in terms of size compared to TCN. It was therefore also discarded.
TCN was chosen due to the fact that it was shown in literature to be performing well on sepsis onset detection and it is in general much lighter than Transformer models.

3.8.1 General TCN Architecture

In this section, the general TCN architecture will be presented.
Fig. 3.4 shows a graphical illustration of the general architecture of the TCN used in this thesis. The details about the depth and size of layers will be discussed for each individual experiment in the respective sections.
As a first step, the TCN takes a window as input. Each channel of the window goes through a different convolutional head, defined as 2 to 4 dilated convolutional blocks consisting of a convolutional layer with dilation 2^n, where "n" is the number of the convolutional block that the convolutional layer belongs to. The convolutional layer in the block is followed by batch-normalization, max-pooling (only in experiment 3.12), and ReLU. Max-Pooling was applied only in the Deployment Model in order to reduce the number of parameters of the model and the size of the activations.
In the next step, the outputs of the convolutional layers for the different heads are then concatenated together and flattened. The size of the flattened array is equal to $sizeOutput * seqLength * nrVitalSigns$, where "sizeOutput" is the size of the output of the TCN for a single vital sign. "seqLength" is equal to the window size in minutes divided by the frequency and further divided by 2 for each max pooling layer (whenever max-pooling is used).
The concatenated and flattened array is then passed to a sequence of dense layer blocks consisting of a dense layer, followed by batch normalization, ReLU, and dropout before being passed to the final dense layer that returns the final output.

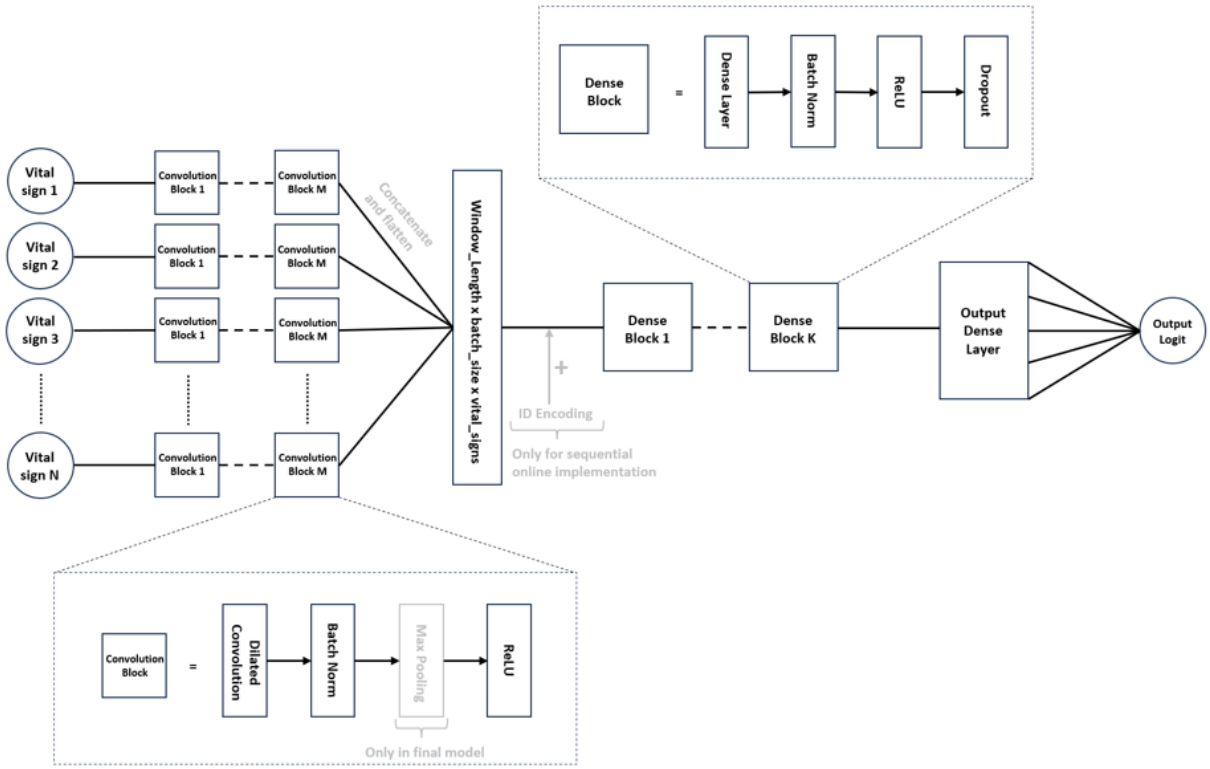

Figure 3.4: Visual Representation of TCN Architecture used in the Thesis

3.9 Experiment 1: Large Windows Model on Online Setting

In this Experiment the Large Windows Model explained in the next subsection is trained and evaluated first with Onset matching and then Without Onset matching. This was done in order to check what effect Onset Matching had on the Large Windows Model in an online setting and to assess how the online implementation of the model would perform.

3.9.1 Model: Large Windows Model

The model for this implementation had 4 dilated convolutional blocks and all convolutional layers had kernel size 5.
The first 1D convolution had size 1x256, while the others were of size 256x256.
The number of dense blocks in the model was 2. The size of the dense layers in each block was (256*360*8)x256 and 256x256 respectively, followed by the last dense layer of size 256x1 that outputs the raw logits. It is important to note that the size of the first

dense block is directly proportional to $dataLengthInMinutes/2$ which in this case (12h data window) was $720/2 = 360$.

The model returned the raw logits since Torch's BCEwithLogits function was chosen instead of the standard Torch's BCE loss function. This is because BCEwithLogits offers more numerical stability compared to a sigmoid followed by BCE.

3.9.2 Sliding Window Implementation

In this experiment, an online setting was used. This also means that instead of having a single static data window for each patient, multiple windows were taken by shifting a window by a predefined 60-minute shift. This can therefore be seen as taking a window and sliding it through the data for each patient until the end of the data or the $onset - predictionTime$ are reached.
Fig. 3.5 shows the sliding window implementation as an example for positive patients or for negative patients that have been assigned an onset according to the method explained in the subsection 3.9.4 dedicated to onset matching when enough data is available.

In case the patient had enough data, the first window taken for the patient started at the beginning of the available data. This is with the exception of positive patients, where an offset was applied if needed in order to ensure that the last window ended exactly 4 hours prior to onset. The windows were then fed to the TCN in a sequential manner where all the windows of a patient were fed to the model before the ones belonging to another patient could start.

In order to allow the model to recognize that multiple windows belonged to the same patient, the patient ID was encoded as a unique combination of sine and cosine and added to the concatenated output of the 8 TCN heads.

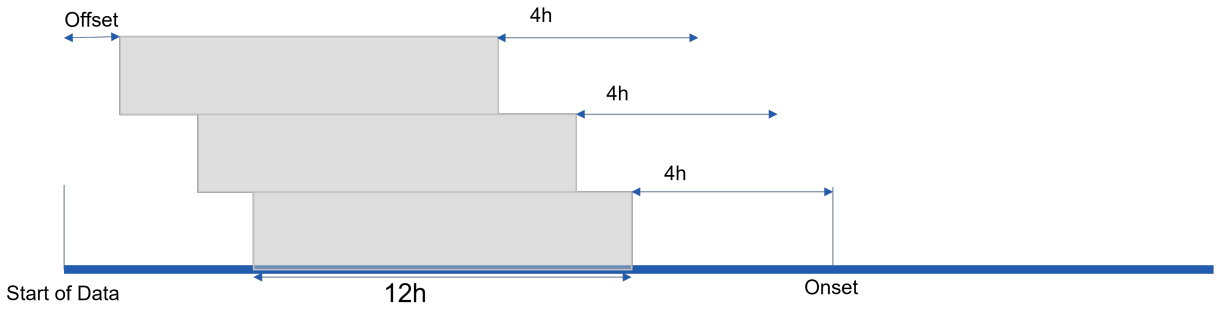

Figure 3.5: Visual Representation of Sliding Window Implementation for Positive Patients or negative patients with matched onset

3.9.3 Padding

For positive patients, it often happened that there was not enough data available. In that case, the starting point of the window was calculated with the following equation: $windowStart = onsetTime - predictionTime - windowLength$, and the missing data was filled with padding using backward filling as shown in Fig. 3.3.

3.9.4 Onset Matching

Onset Matching is a technique where an "artificial" onset time is assigned to negative patients during preprocessing. This is done in order to balance the amount of data that the model sees for individual positive and negative patients.
There are various ways in which Onset Matching can be done. The first technique is random Onset Matching where a random onset time is assigned to negative patients. Another Method is relative Onset Matching where the control cases are paired with positive patients and they are assigned an onset time that has to have the same ratio of $patientDataAvailable/(onsetTime - dataStartingTime)$ as the one of the positive patient it was paired with. A third option is absolute Onset Matching. This is the method used in this experiment.
Absolute Onset Matching was implemented by randomly pairing each control case in the undersampled subset to a positive case for each epoch. The negative sample would then be assigned an onset time so that its admission-to-sepsis time would be equal to one of the positive patients it was paired with.

Fig. 3.6 shows a visual representation of the Onset Maching Implementation.

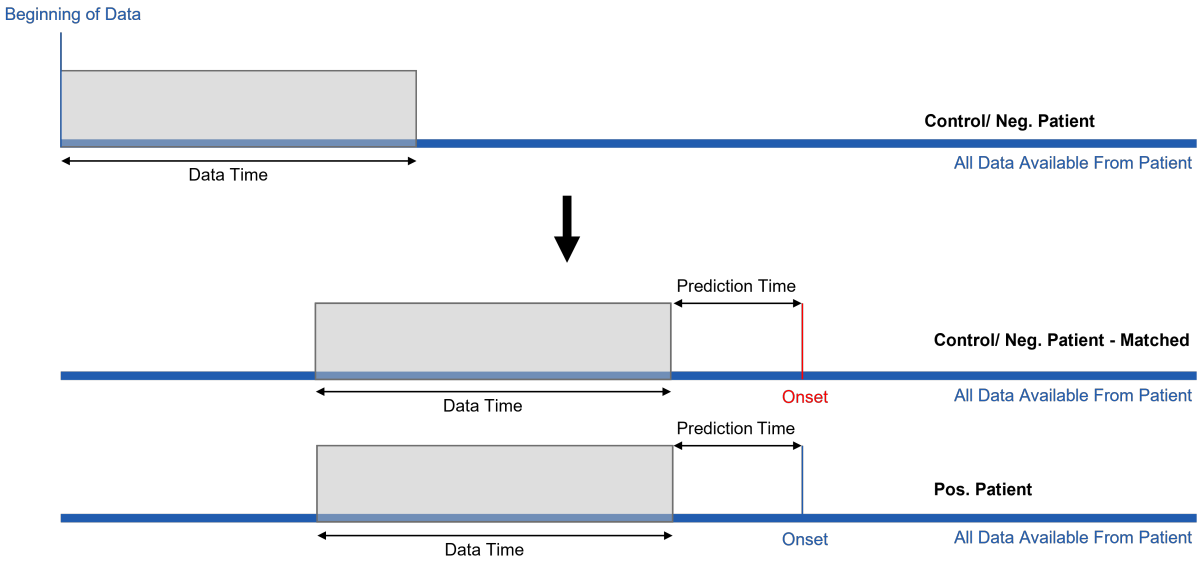

Figure 3.6: Visual Representation of Onset Matching Implementation

3.10 Experiment 2: Horizon Evaluation with Large Windows Model

In this experiment, the model explained in 3.9.1 is used on an offline setting with horizon evaluation. This was done in order to assess the change in performance when the same model is trained in a different setting.

3.10.1 Model: Large Windows Model

For this experiment, the same model was used as in 3.9. However, the setting switched from an online to an offline setting. Details are given in the following subsection.3.10.2.

3.10.2 Horizon Evaluation Implementation

Horizon Evaluation refers to training and evaluating a model using different prediction times.
In this experiment, a horizon evaluation with different window sizes and prediction times was done in order to assess the way these two factors influence the performance of a model.
The window sizes were 4h, 8h, and 12h and the prediction times were 4h, 8h, 12h, and 16h.
For control cases, the window started at the time of admission and ended at $admissionTime+$

$windowSize$ while for positive cases the window ended at $onsetTime - predictionTime$ and started at time $onsetTime - predictionTime - windowSize$.

3.10.3 Padding

As for the previous experiment 3.9, for positive patients, it often happened that there was not enough data available. Padding was therefore required in order to keep the size of the data fed to the model consistent. The exact method used to implement padding can be found in a previous subsection 3.9.3.

3.11 Experiment 3: Horizon Evaluation with Small Windows Model

In this experiment the Small Windows Model explained in the next subsection is evaluated with horizon evaluation using different window sizes such that $windowSize + predictionTime \leq 8$ (i.e. no padding was done).
The goal of this experiment was therefore to assess the impact of padding on a model's performance with respect to different window sizes and prediction times.

3.11.1 Model: Small Windows Model

The model for this implementation had 4 dilated convolutional blocks and all Convolutional Layers had kernel size 7. The first 1D Convolution had size 1x32 while the others were of size 32x32.
The number of dense blocks in the model was 2. The size of the dense layers in each block was (32*windowSize*8)x64 and 64x64 respectively, followed by the last dense layer of size 64x1 that outputs the raw logits.
The Small Windows Model's parameters and activations are of type float32 (i.e. not quantized) and the model's size is approximately 7.2MB.

3.11.2 Horizon Evaluation Implementation

In this experiment, as in the previous one 3.10, a horizon evaluation with different window sizes was done in order to assess the way these two factors influence the performance of a model. However, the window sizes used in this case were 2h, 3h, and 4h. The prediction times were 1h, 2h, 3h, and 4h.
The details of the exact implementation are explained in section 3.10.

3.12 Experiment 4: Horizon Evaluation with Deployment Model

In this experiment, a lightweight model for MCU Deployment implemented on Tensorflow is evaluated in an offline setting using horizon evaluation.
The Tensorflow implementation was built from scratch with the goal of reproducing the model presented in experiment 3 3.11 in order to be optimized for size. The Tensorflow implementation was necessary for two reasons: First, it was a useful confirmation of the results achieved in experiment 3. Second, it made it easier to prepare the model for exportation on MCU thanks to Tensorflow's TFLite for MCU.
The main optimizations done in order to reduce the size of the model previously obtained were the removal of two biomarkers, the introduction of Max-Pooling layers, and quantization.
The end result was the model explained in the next subsection.

3.12.1 Model: Deployment Model

For this experiment, two vital signs were dropped, namely mbp and blood glucose. Mean blood pressure was removed since it did not offer any new information about the state of the patient, while blood glucose was dropped due to the fact that there was not enough data for this biomarker.
This model had 4 Dilated Convolution Blocks. All convolutional layers had kernel size 3. The first Convolutional Layer had size 1x32 while the other Convolutional Layers had size 32x32. Max pooling with kernel size 2 was used in all blocks for the experiment with a frequency of 2min but it was not used for the low-granularity experiment since no further reduction in input size was needed.
Furthermore, there were 2 Dense Blocks of size $seqLen * 6 * BatchSize$x32 and 32x32 respectively, where seqLen is the length of the window following all the Convolutional Blocks. "seqLen" is, therefore, $floor(120/2^4) = 7$ for the high-granularity experiment and equal to 4 for the low-granularity experiment.
The output is then given by a dense layer of size 32x1 followed by a sigmoid. Therefore on Tensorflow, contrary to what was done on Torch, the model returned the probability that a patient will develop sepsis in the next "prediction time" hours instead of the logit.

The model size is around 120KB using the high-granularity dataset and approximately 100KB using the downsampled dataset. It has a maximum activation size of 3.75KB and 0.75KB using the original and the downsampled datasets respectively.

3.12.2 Horizon Evaluation Implementation

Although the model greatly differs from section 3.11, the horizon evaluation is identical between the two experiments.

3.12.3 Downsampling to Low Granularity

The experiment was done twice: the first time it was done with the original HiRID high-granularity dataset. The second time it was done by taking the vital signs from HiRID and downsampling them to a 1-hour granularity by taking the first value each hour in order to assess how the model would perform with a low-granularity dataset. It is important to take the first value instead of the mean over the hour for the downsampling since, as shown in Fig. 4.7, the model performance is influenced by taking the mean over the hour. This is because by taking the mean, the information inherent to the higher granularity of the original dataset is extracted.

3.12.4 Quantization

The Deployment Model was quantized by doing full-integer-quantization.
With full-integer-quantization, all the model's parameters together with the activations, the input, and the output are reduced from a 32-bits float into an 8-bits integer, thereby decreasing both the memory consumption for both FLASH and RAM 4-fold.

3.13 Search For Optimal Model to be Deployed

In order to find a model with a relatively small number of parameters that also performed well in terms of AUROC, AUPRC, and Accuracy, Random Search was used.
Random Search was chosen instead of other methods, such as Bayesian Search, which aims at optimizing one metric. This is because by optimizing for AUROC, most models would have been too big to fit in the MCU, and by optimizing for model size, not many models that showed a decent performance would have been found. To back up this claim, fig. 4.8 shows the correlation between size and performance. The parameters that were given to the search are:

- Added Noise: True, False
- Batch Size: 16, 32, 64, 128, 256
- Convolutional Layers (length of array shows depth and number shows output size): [16, 16], [32, 32], [64, 64], [64, 64, 64, 64], [32, 32, 32, 32], [16, 16, 16, 16].
- Dense Layers: [16], [32], [64], [16, 16], [32, 16], [64, 16], [64, 64], [32, 32], [64, 32]

- Epochs: in the range [10, 20]
- Convolutional Layer Kernel Size: 3, 5, 7, 9
- Max Pooling (Length of the array depends on depth of TCN. 2=Use Max Pooling with kernel size 2 in ith layer, 0=Do not Use Max Pooling in ith layer): [2, 2], [0, 0], [2, 0], [0, 2], [2, 2, 2, 2], [0, 0, 0, 0], [2, 0, 2, 0], [0, 2, 0, 2]
- Sensor fusion in TCN: True, False
- Single TCN: True, False

3.14 Sensor Fusion

Sensor Fusion refers to the combination of information coming from different sources (or sensors). The TCN explained in the section above 3.8.1 already implements Sensor Fusion but only on the dense layers, which take the concatenated outputs of the 8 heads of the TCN as an input. However, passing multiple channels to the same convolutional head could allow the model to extract even more useful features that stem from correlations between pairs of different biomarkers.

In this thesis, Sensor Fusion at the convolutional layers was implemented by feeding the HR channel and the sbp channel together into a single head of the TCN while leaving the other channels independent from each other.

Chapter 4

Results

4.1 Experiment 1: Large Windows Model on Online Setting

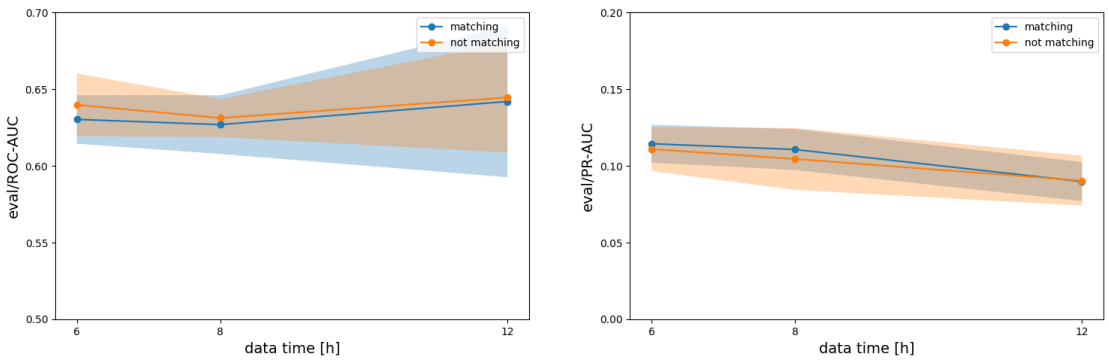

Figure 4.1: Performance of Model in Sequential Online Setting for window sizes in range $[6, 12]$ with and without Onset Matching. 3-Fold Cross-Validation was done. Mean and std are displayed.

From fig. 4.1, it can be seen that Onset Matching did not impact the AUROC and the AUPRC since an almost perfect overlap between the two lines can be observed. In fact the absolute value of the mean difference in AUROC and AUPRC between the 2 lines is around 0.0055 for AUROC and around 0.0030 for AUPRC.
It can also be seen that the metrics are stable over different window sizes with a mean AUROC value of around 0.633 and a mean AUPRC value of around 0.105.

Fig. 4.2, which displays the confusion matrix for the two evaluations, however, shows

that onset matching increases the TP by 1.279% and the FP by 20.178%, while decreasing the TN by 20.178% and FN by 1.279%, compared to when onset matching was not used.

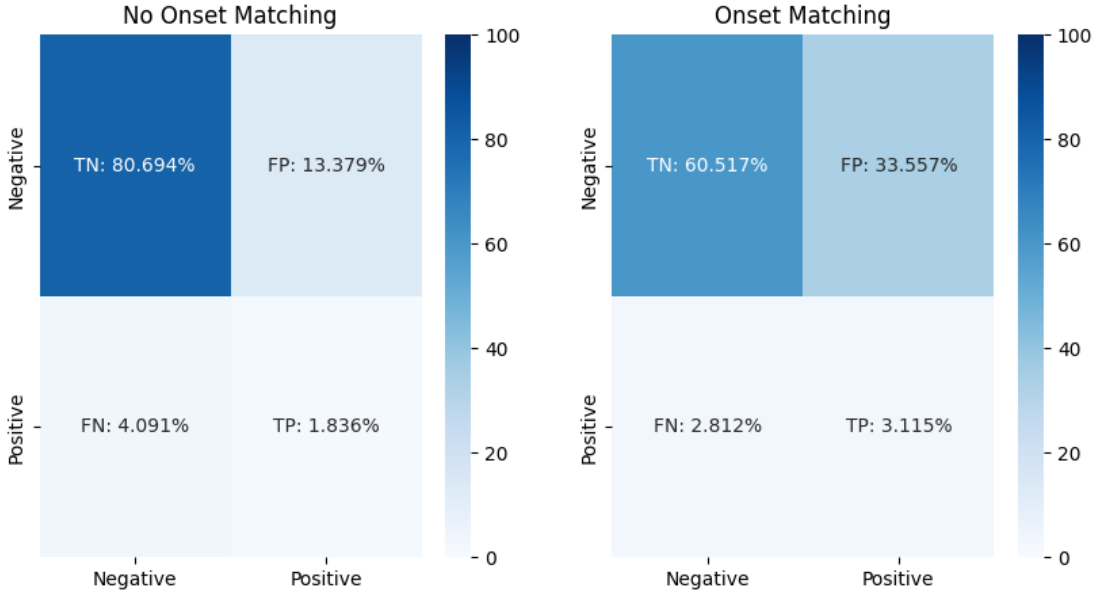

Figure 4.2: Confusion Matrix Showing the percentage of true negatives, true positives, false negatives, and false positives for the evaluation of the model trained with and without onset matching

4.2 Experiment 2: Horizon Evaluation with Large Windows Model

Fig. 4.3 shows 3 columns of graphs with different window sizes ranging from 4 to 12 hours. The top row shows the AUROC while the bottom row shows the AUPRC.
It can be seen that AUROC and AUPRC increase with increasing prediction time for large data windows.
For a Data Time of 4 hours, it can be seen that both the AUROC and the AUPRC decrease when the prediction time is increased from 4 hours to 8 hours. However when the prediction time is further increased, both performance metrics follow the increasing trend that can also be seen with a Data Time of 8 hours and 12 hours.
It is important to note that when both Data Time and Prediction Time are 4 hours, it is the only instance where no padding was applied. This is because given the selected inclusion criteria 3.3, it is ensured that when the condition $predictionTime + dataTime <= 8hours$ is fulfilled, then enough data is available for each positive and negative patient.

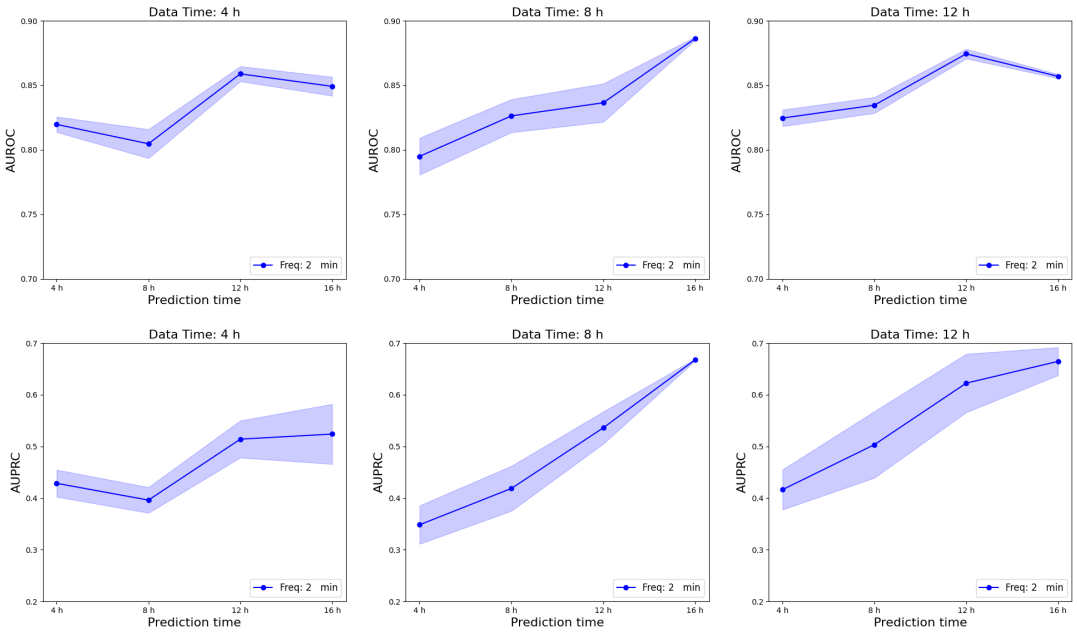

Figure 4.3: Mean AUROC and AUPRC for 3-Fold Cross-Validation. Window lengths in the range $[4h, 12h]$ and prediction times in the range $[4h, 16h]$. 95% confidence interval is displayed around the mean.

The AUROC and AUPRC when the condition is fulfilled are around 0.825 and 0.43 respectively.

4.3 Experiment 3: Horizon Evaluation with Small Windows Model

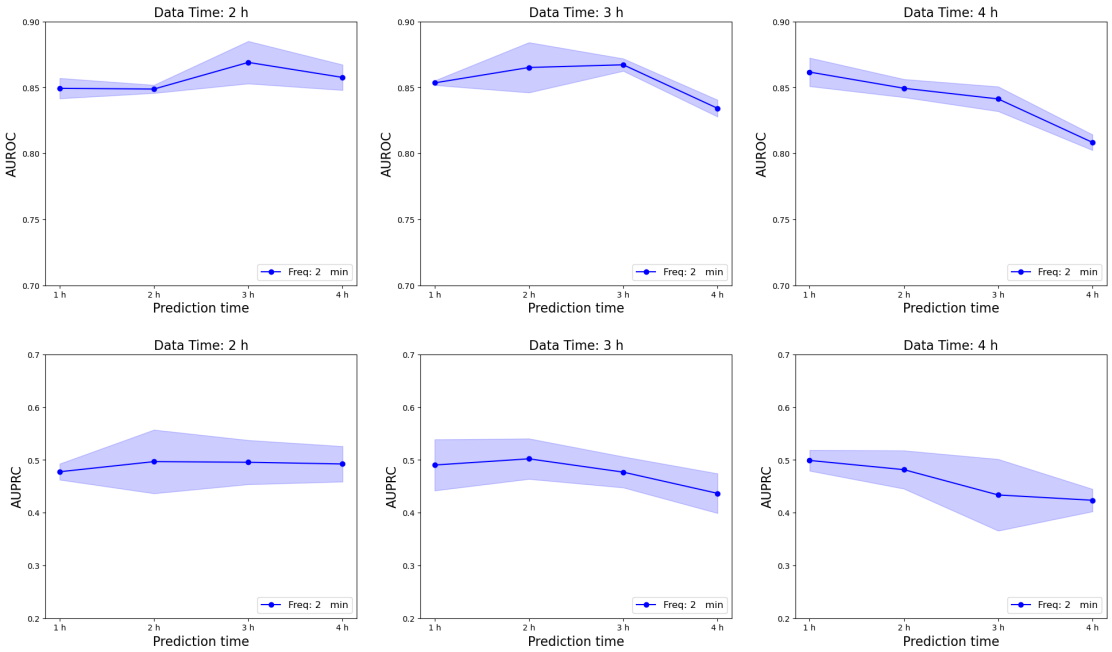

Figure 4.4: Mean AUROC and AUPRC for 3-Fold Cross-Validation. Window lengths in the range $[2h, 4h]$ and prediction times in the range $[1h, 4h]$. 95% confidence interval is displayed around the mean.

Fig. 4.4 shows 3 columns of graphs with different window sizes ranging from 2 to 4 hours all plotted against a prediction time ranging from 1 to 4 hours. The top row shows the AUROC while the bottom row shows the AUPRC.

It can be seen that for data windows and prediction times such that $dataTime + predictionTime \leq 8hours$, AUROC and AUPRC decrease with increasing prediction time.
For all combinations of prediction time and Data Time, the AUROC is > 0.8 and the AUPRC is > 0.4.
The model performs particularly well with a Data Time of 2 hours, as shown by the AUROC of around 0.85 and the AUPRC of around 0.5 for all prediction times.
Generally, the performance of the model with the 3 different window sizes starts at about the same value for both AUROC and AUPRC with a prediction time of 1 hour but the rate of decrease in performance gets higher when the Data Time is increased from 2 hours up to 4 hours.

It is important to note that due to the fact that the sum of data time and prediction time is always less than 8 hours, all patients have enough data due to the chosen inclusion criteria 3.3, hence no padding is applied.

4.4 Experiment 4: Horizon Evaluation with Deployment Model

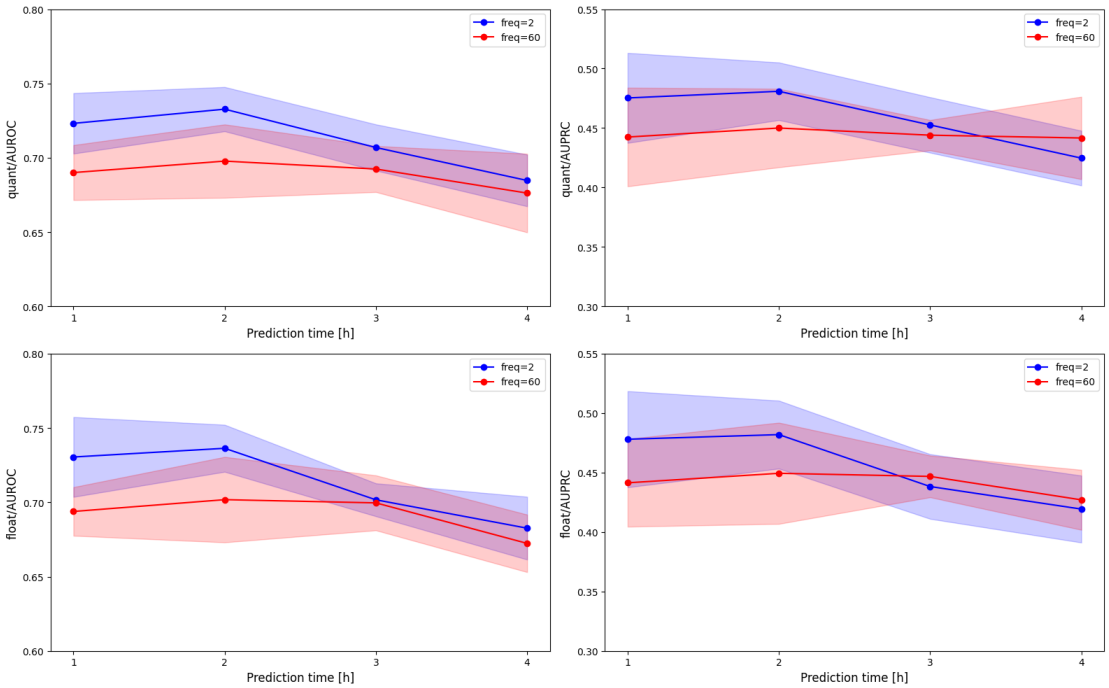

Figure 4.5: AUROC, AUPRC and Accuracy of model Fully Quantized (8bits) model compared to same model with Float (32bits) parameters and activations with a 5-fold Cross-Validation. Mean and std are displayed.
Both the fully quantized and the float models have also been trained with a version of the HiRID dataset whose granularity has artificially been reduced to 1 hour by taking the first value seen each hour.

Fig. 4.5 shows 4 graphs. In the first column, AUROC is displayed while in the second one, AUPRC is presented. The first row shows the quantized model's graphs and the second row displays the pre-quantization (float) model's performance.
In all plots, the performance of the models trained and evaluated on a version of the HiRID dataset that has been downsampled 3.12.3 to 60 minutes is also shown (corresponding to the red line in the plots).
The figure shows that there is no significant change in performance after quantization. In fact, the mean difference between the quantized model and the float model over all

the points is 0.0046 for AUROC and 0.0056 for AUPRC. The figure also shows that the model's performance gets slightly worse by reducing the granularity of the dataset to 60 minutes.

4.5 Additional Results

In this section, results are shown that motivate some choices that are explained in the Methods chapter 3.

4.5.1 Normalization Approaches

Fig. 4.6 shows the performance of the Deployment Model from experiment 3.12 using data on which the Min-Max Scaler has been used in two different ways. The blue line shows the performance of the model where the data had been scaled by fitting the scaler on the whole dataset, while the red line shows the performance of the model where the scaler was newly fitted for each batch of size 32.
In the first column, the AUROC is shown, while in the second column the AUPRC is displayed. The first row shows the performance metrics of the quantized model while the second row shows the metrics of the same model but with float32 parameters and activations.

It can be seen that even with a relatively small batch size as the one used in this experiment, applying the scaler for each batch, as was done in experiments 1 to 3 doesn't have a big impact on the model's performance. In fact the mean difference between the two lines is around 0.0035 for the AUROC (quant and float) and around 0.01 for the AUPRC (quant and float).
It can also be seen that the difference in performance between the two scaling modalities becomes smaller and smaller with increasing prediction time and at a prediction time of 4 hours, the two lines almost overlap.

One noticeable difference though is that the standard deviation is greatly reduced when the same transformation is applied to all samples (blue line) instead of applying a different scaler for each batch (red line).

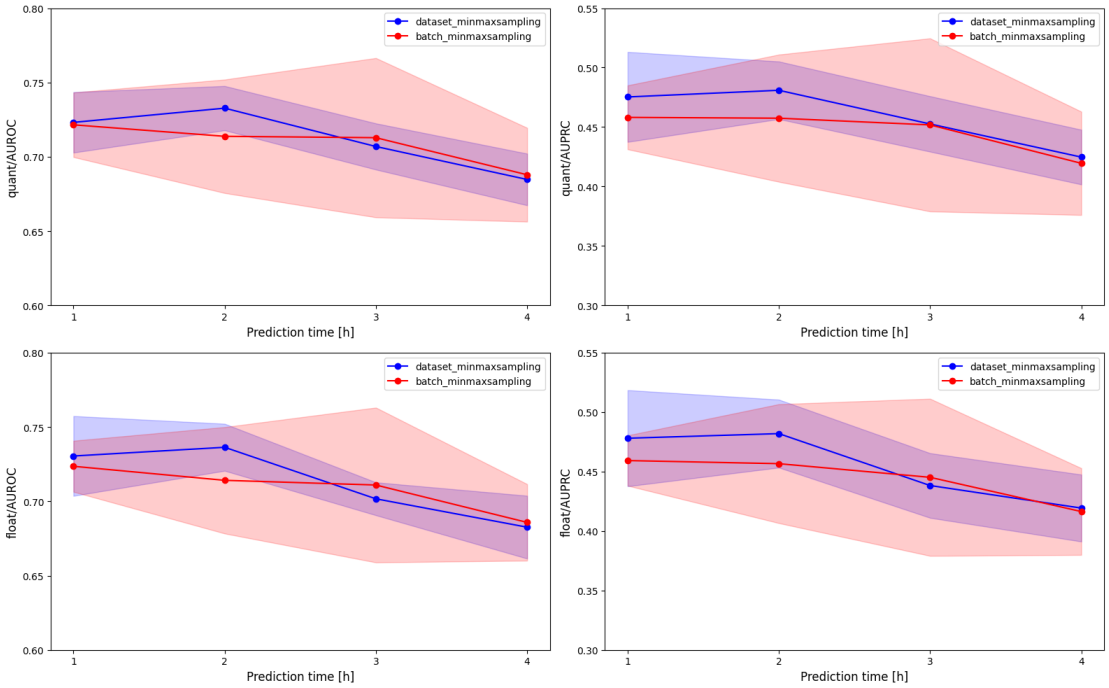

Figure 4.6: This figure shows the impact in performance that fitting the min-max-sampler over each individual batch instead of fitting it over the dataset has. We can see a small decrease in AUPRC and AUROC but accuracy seems to be slightly better. Overall no big difference is observed. The batch size used was 32. By increasing the batch size an even smaller difference should be observed. The model used to get this data can be found in the following section 3.12

4.5.2 Undersampling Approaches

Fig. 4.7 shows the performance of the Deployment Model 3.12 trained and tested using two variations of the undersampled HiRID dataset.
The purple line shows the performance of the model on a dataset that was undersampled to a 1-hour frequency by taking the mean of the original data over the hour. For the yellow line on the other hand, the undersampling was done by taking the first value available each hour.
In the first column, the AUROC is shown, while in the second column the AUPRC is displayed. The first row shows the performance metrics of the quantized model while the second row shows the metrics of the same model but with float32 parameters and activations.

It can be seen that undersampling by taking the mean generally results in a better overall performance of the model even though not by a large margin. The mean difference over the two lines is around 0.02 for both AUROC and AUPRC.

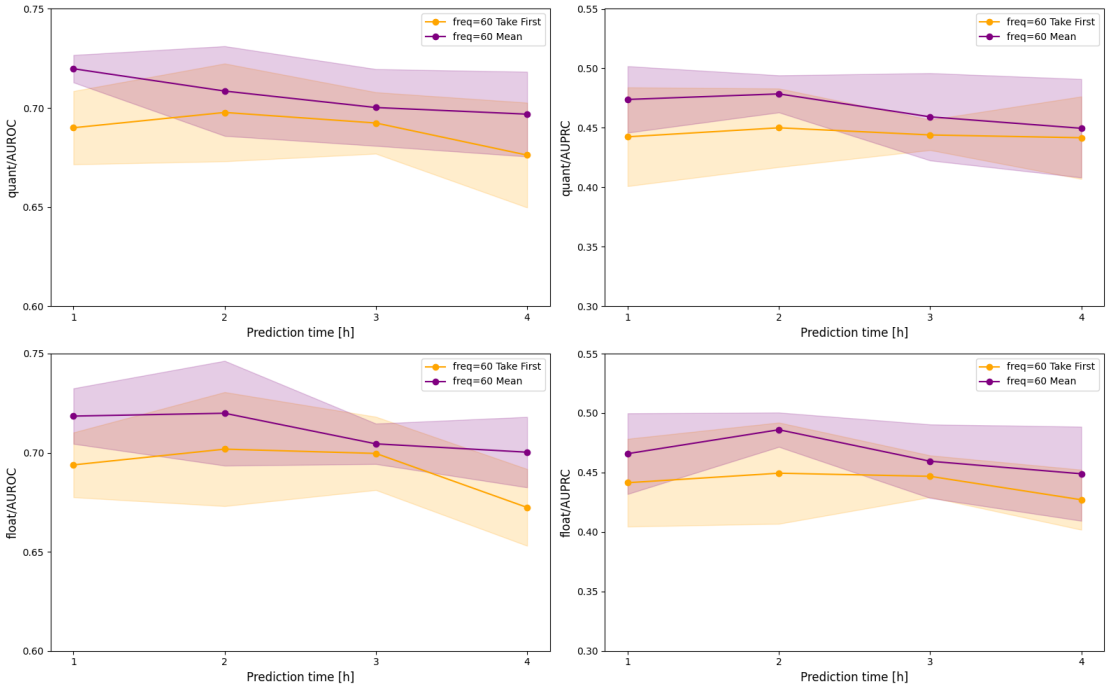

Figure 4.7: This figure compares two approaches to undersampling the HiRID dataset to a frequency of 60 minutes. The data shown is the result of a 5-Fold Cross-Validation. Mean and std are displayed. The first approach is to take the first value every 60 minutes while the other approach is to take the mean over the 60 minutes. One can observe that although AUPRC and AUROC stay relatively similar, the accuracy shows an increase in performance by taking the mean over the hour. This means that by taking the mean we are making use of information extracted from the 2 minutes granularity of the HiRID dataset, therefore, for a more accurate estimate of the performance of the model on datasets with a granularity of 1 hour, it is best to use the first approach. The model used to get this data can be found in the following section 3.12

4.5.3 Random Search

Fig. 4.8 shows the AUROC (left) and AUPRC (right) of the quantized models tested during the Random Search.

A general tendency of increase in performance across all metrics can be observed with increasing model size. This is highlighted by the regression lines for both AUROC and AUPRC.

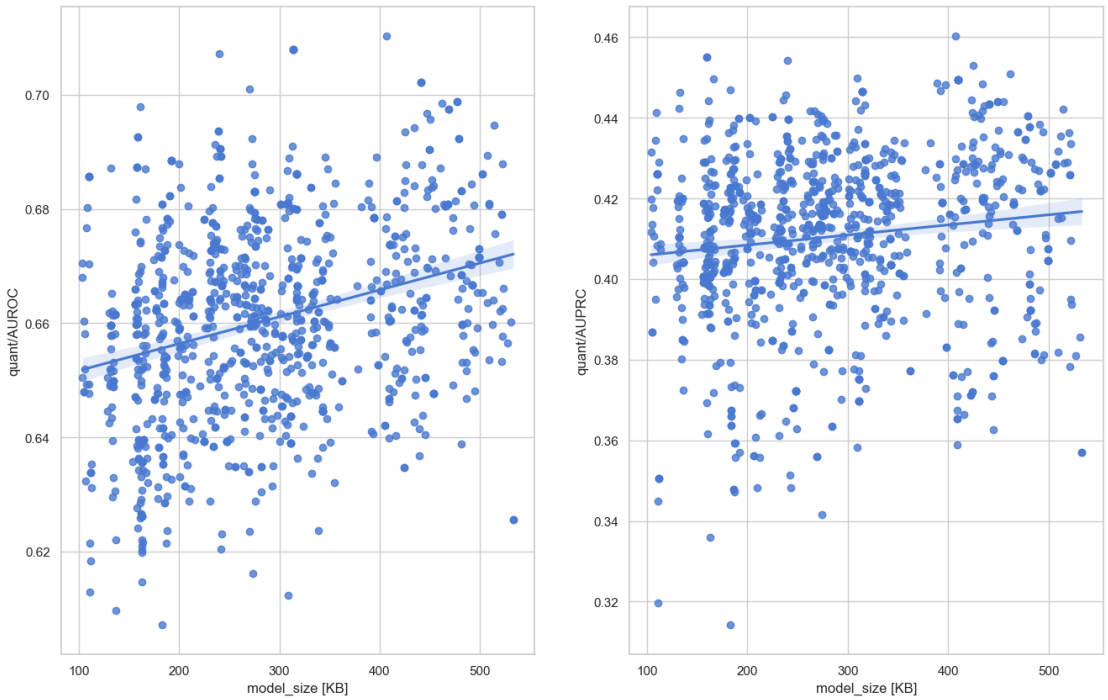

Figure 4.8: This figure Displays the performance of the models evaluated in the random search over an increasing number of parameters. The regression line shows the improvement in the various performance metrics with increasing model size

Chapter 5

Discussion

In fig. 4.3 it can be seen that the performance of the Large Windows Model goes up with an increase in the prediction time from 4h to 16h. This is because, with big data windows, padding is the main factor in determining the performance of the model, since contrary to what is expected, the task becomes increasingly easier for the Large Windows Model as the prediction time gets further away from the onset due to the fact that more and more of the positive patients' windows are padding. Fig. 4.4, on the other hand, shows the expected behavior, which is a decline in the performance metrics with increasing prediction time. This is because, with a prediction time and window size such that $predictionTime + windowSize <= 8h$, no padding was added. This result further shows that the Small Windows Model is achieving good results without the help of padding. Although the Small Windows Model achieved a good performance, it was still too heavy for deployment on an MCU.

Regarding the sequential online implementation, fig. 4.1 shows no difference in performance between using Onset Matching and not using it. This means that the Large Windows Model in the sequential online setting is not working. If it worked, a similar behavior to 4.3 would be expected for the Large Windows Model without onset matching. This is because, as discussed above, without Onset Matching, the Large Windows Model should have been able to distinguish positive and negative patients just by looking at the padding. When Absolute Onset Matching was used, on the other hand, the Large Windows Model was not able to distinguish positive and negative patients just by looking at padding. If the Large Windows Model worked on the Sequential Online setting, it would therefore be expected to perform much worse when Onset Matching is used than when Onset Matching is not used.

From 4.5 we can see that quantization had no impact on the performance of the Deployment Model, with the benefit of reducing the model's size 4-fold. The figure also compares the performance of the Deployment Model trained and tested with 2-minute

and 60-minute granularity. This also demonstrates that the granularity of the dataset had a minimal effect on the performance. The fact that the Deployment Model's performance is stable with a reduction in granularity does not mean that the model can generalize well and maintain its performance with an external dataset that has lower granularity such as MIMIC-III. This is because the low-granularity version of HiRID is still sampled from the same distribution as the standard high-granularity dataset. Train and evaluation with an external dataset are therefore required in order to have a good estimate of the real quality of the Deployment Model.
As expected, by increasing the prediction time, the performance declined. However, with a 4-hour prediction time, the Deployment Model still had an AUROC of approximately 0.7 and an AUPRC of approximately 0.45 which considering its size of a little more than 100KB, can be considered an important achievement. Fig. 4.7 shows that taking the mean over the hour increases the performance compared to taking the first value each hour. This is likely because using the mean over 60 minutes for the downsampling confers advantages that are derived from the high granularity of the underlying dataset. This could therefore be a useful technique to reduce the input size, thereby also reducing a model's number of parameters and max activation size.

Chapter 6

Conclusion and Future Work

In this thesis, I explored the literature behind the difficult and fascinating problem of sepsis early onset detection. Along the way, I explored the limitations of big input sizes and various artifacts introduced during data pre-processing. The end result of my thesis is a good lightweight model capable of being deployed on an MCU, which is an achievement for the research on this topic since having a lightweight model with good performance is of utmost importance in order to go from research to deployment in a hospital setting. Although the Deployment Model performs fairly well, the parameter space could be explored more extensively. Searching better through the parameter space is therefore a relatively easy thing to do that could greatly improve the performance-to-size ratio of the Deployment Model. Moreover, Sensor Fusion at the convolutional part of the network was not thoroughly explored. It would therefore be interesting to assess the correlation between different biomarkers in order to combine them in a meaningful manner in one head of the TCN. Validation on an external dataset such as MIMIC-III is also an important next step as a means to evaluate whether the Deployment Model can generalize well across different data distributions, which is critical if the model is to be deployed in clinical settings in the future. Finally, unsupervised or semi-unsupervised techniques would be interesting to explore, especially in order to avoid the circularity problem mentioned in a review by Moor et. al. [5]. Additionally, my thesis focused more on Machine Learning and on optimization techniques to reduce the number of parameters of a model. For this reason, the work of this thesis could provide a basis for MCU implementation which could be the central part of someone else's project.
To conclude I consider this thesis a success, not only for the end result but especially for everything that I was able to learn in the past 3 months.

Bibliography

[1] WHO, "Sepsis," 2020, accessed Jun. 05, 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/sepsis

[2] W. H. Organization, "Who calls for global action on sepsis - cause of 1 in 5 deaths worldwide," 2020, accessed Jun. 05, 2023. [Online]. Available: https://www.who.int/news/item/08-09-2020-who-calls-for-global-action-on-sepsis---cause-of-1-in-5-deaths-worldwide

[3] P. E. Marik and A. M. Taeb, "SIRS, qSOFA and new sepsis definition," *Journal of Thoracic Disease*, vol. 9, no. 4, pp. 943–945, Apr. 2017. [Online]. Available: https://doi.org/10.21037/jtd.2017.03.125

[4] A. Johnson, T. Pollard, and R. Mark, "Mimic-iii clinical database," Sep 2016. [Online]. Available: https://physionet.org/content/mimiciii/1.4/

[5] M. Moor, B. Rieck, M. Horn, C. R. Jutzeler, and K. Borgwardt, "Early prediction of sepsis in the icu using machine learning: A systematic review," *Frontiers in Medicine*, vol. 8, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fmed.2021.607952

[6] Z. Wang and B. Yao, "Multi-branching temporal convolutional network for sepsis prediction," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 2, pp. 876–887, 2022.

[7] M. Moor, M. Horn, B. Rieck, D. Roqueiro, and K. Borgwardt, "Early recognition of sepsis with gaussian process temporal convolutional networks and dynamic time warping," 2019.

[8] R. Ding, Y. Zhou, J. Xu, Y. Xie, Q. Liang, H. Ren, Y. Wang, Y. Chen, L. Wang, and M. Huang, "Cross-hospital sepsis early detection via semi-supervised optimal transport with self-paced ensemble," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 6, pp. 3049–3060, 2023.

[9] M. A. Reyna, C. S. Josef, R. Jeter, S. P. Shashikumar, M. B. Westover, S. Nemati, G. D. Clifford, and A. Sharma, "Early prediction of sepsis from clinical data," *Critical Care Medicine*, vol. 48, no. 2, pp. 210–217, Feb. 2020. [Online]. Available: https://doi.org/10.1097/ccm.0000000000004145

[10] S. Liu, B. Fu, W. Wang, M. Liu, and X. Sun, "Dynamic sepsis prediction for intensive care unit patients using XGBoost-based model with novel time-dependent features," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 8, pp. 4258–4269, Aug. 2022. [Online]. Available: https://doi.org/10.1109/jbhi.2022.3171673

[11] T.-C. Do, H.-J. Yang, G.-S. Lee, S.-H. Kim, and B.-G. Kho, "Rapid response system based on graph attention network for predicting in-hospital clinical deterioration," *IEEE Access*, vol. 11, pp. 29 091–29 100, 2023.

[12] T. Desautels, J. Calvert, J. Hoffman, M. Jay, Y. Kerem, L. Shieh, D. Shimabukuro, U. Chettipally, M. D. Feldman, C. Barton, D. J. Wales, and R. Das, "Prediction of sepsis in the intensive care unit with minimal electronic health record data: A machine learning approach," *JMIR Medical Informatics*, vol. 4, no. 3, p. e28, Sep. 2016. [Online]. Available: https://doi.org/10.2196/medinform.5909

[13] D. Wang, J. Li, Y. Sun, X. Ding, X. Zhang, S. Liu, B. Han, H. Wang, X. Duan, and T. Sun, "A machine learning model for accurate prediction of sepsis in ICU patients," *Frontiers in Public Health*, vol. 9, Oct. 2021. [Online]. Available: https://doi.org/10.3389/fpubh.2021.754348

[14] Q. Mao, M. Jay, J. L. Hoffman, J. Calvert, C. Barton, D. Shimabukuro, L. Shieh, U. Chettipally, G. Fletcher, Y. Kerem, Y. Zhou, and R. Das, "Multicentre validation of a sepsis prediction algorithm using only vital sign data in the emergency department, general ward and icu," *BMJ Open*, vol. 8, no. 1, 2018. [Online]. Available: https://bmjopen.bmj.com/content/8/1/e017833

[15] M. Moor, N. Bennet, D. Plecko, M. Horn, B. Rieck, N. Meinshausen, P. Bühlmann, and K. Borgwardt, "Predicting sepsis in multi-site, multi-national intensive care cohorts using deep learning," 2021.

[16] H. Yèche, A. Pace, G. Rätsch, and R. Kuznetsova, "Temporal label smoothing for early event prediction," 2022.

[17] M. Faltys, M. Zimmermann, X. Lyu, M. Hüser, S. Hyland, G. Rätsch, and T. Merz, "Hirid, a high time-resolution icu dataset," 2021. [Online]. Available: https://physionet.org/content/hirid/1.1.1/

[18] P. E. Marik and A. M. Taeb, "Sirs, qsofa and new sepsis definition," *Journal of Thoracic Disease*, vol. 9, no. 4, 2017. [Online]. Available: https://jtd.amegroups.org/article/view/12738

[19] S. Nemati, A. Holder, F. Razmi, M. D. Stanley, G. D. Clifford, and T. G. Buchman, "An interpretable machine learning model for accurate prediction of sepsis in the ICU," *Critical Care Medicine*, vol. 46, no. 4, pp. 547–553, Apr. 2018. [Online]. Available: https://doi.org/10.1097/ccm.0000000000002936