NANYANG TECHNOLOGICAL UNIVERSITY

TIME SERIES PREDICTIONS IN HEALTHCARE APPLICATIONS

Zhong Shaojie

School of Computer Science and Engineering

2024

NANYANG TECHNOLOGICAL UNIVERSITY

SCSE23-0702

TIME SERIES PREDICTIONS IN HEALTHCARE APPLICATIONS

Submitted in Partial Fulfilment of the Requirements
for the Degree of Bachelor of Science in Data Science & Artificial Intelligence
of the Nanyang Technological University

by

Zhong Shaojie

School of Computer Science and Engineering

2024

Abstract

In recent years, deep learning has been applied in the medical field to perform at sorts of tasks such as the prediction of in-hospital mortality. Due to the inconsistency of follow-up appointments and recordings of vital signs, irregularity is inherently present in medical data. This irregularity inherent in the data poses substantial hurdles to traditional machine-learning techniques as these techniques can only be employed on time-series data with regular intervals. Over the years, researchers have devoted considerable efforts to tackling this challenge which has led to the development of methods falling into two main categories: interpolation-based and non-interpolation-based models. In this study, we propose a novel approach, STraTS-mTAND, which integrates techniques from STraTS, a non-interpolation model, and mTAND, an interpolation model, to address the binary in-hospital mortality classification problem using irregularly sampled time-series data. Our approach leverages the strengths of each technique while mitigating their respective limitations. We evaluate the effectiveness of STraTS-mTAND using the PhysioNet Challenge 2012 dataset and the MIMIC-III dataset. Our experimental results demonstrate that our approach outperforms existing methods in both PR-AUC and ROC-AUC. Additionally, our approach has also shown to perform better with lesser training data and with sparser and more irregular time-series. Furthermore, we analysed the PhysioNet Challenge 2012 dataset to provide valuable insights, which may be used to improve medical resource allocation for patients in the ICUs.

Acknowledgments

I would like to thank Assistant Professor Fan Xiuyi from Lee Kong Chian School of Medicine for this opportunity to conduct the research as well as the advice, encouragement and support throughout this project. Without his valuable and motivational feedback at weekly meetings, this research paper would never have been completed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Electronic Health Record (EHR) is a system that stores patients information in a secure database, allowing patients information to be readily accessible by any authorised users. It stores data ranging from demographics, medical diagnosis, prescription and billing information. In recent years, Electronic health records(EHRs) adoption has steadily increased, resulting in a wealth of data [1].

A subset of the data collected through EHRs comes from the Intensive Care Units (ICU). Patients who are admitted to the ICU are usually in life-threatening conditions. The medical decisions made in the first few hours after admission are critical and are often more prone to error than the decisions made in later time [2]. Hence automated real-time predictions can be extremely beneficial in assisting clinicians in making medical decisions. By leveraging these predictions, clinicians can make more informed decisions, improve patient outcomes, and mitigate the risk of errors.

The data collected in the ICU are mainly multivariate irregularly sampled time-series data. These data are often used in tackling medical prediction problems such as mortality prediction, phenotype classification and length of hospitalisation [3].

There are a few distinct properties of multivariate irregularly sampled time-series data as shown in figure 1.1

1. The time intervals between observations are not regular.

2. The observation timings are misaligned across variables, meaning that not all variables are recorded at the same timestamp.

3. It is possible for a time-series to not record any observation for a variable.

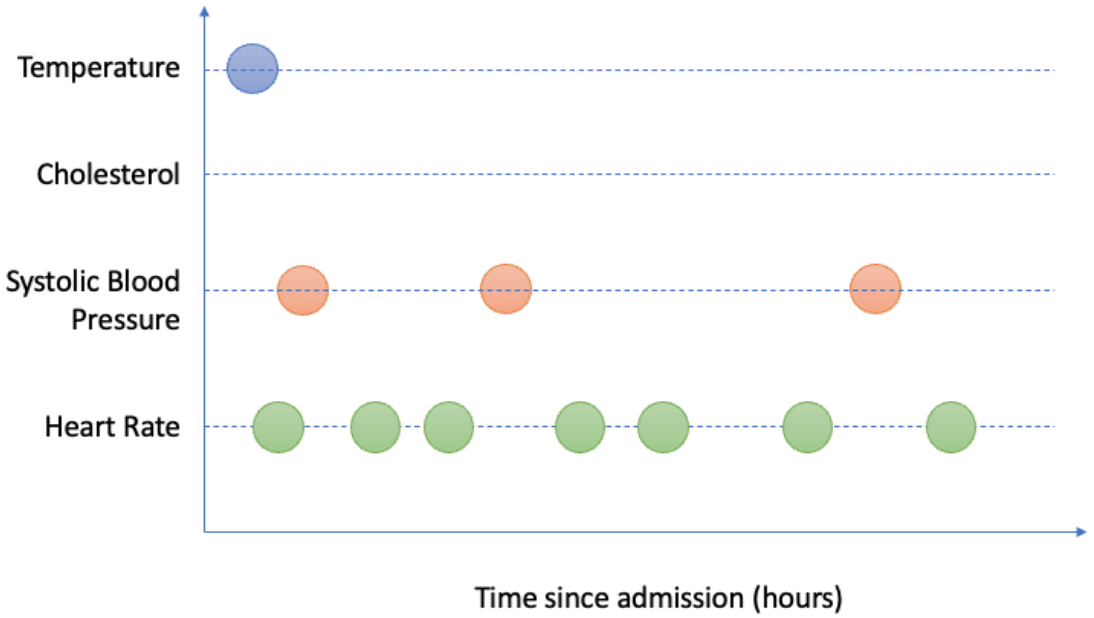

Figure 1.1: Example of multivariate irregularly sampled time-series in healthcare setting

Due to the highly irregular and sparse nature of the observations, it is challenging to utilise traditional machine learning models that require a time-series of regular time intervals. The standard method for dealing with irregularly sampled time-series utilises imputations to transform it into a regular time-series [4] through techniques such as forward-filling or mean-imputation. However, this can introduce bias and reduce the accuracy [5].

To address the limitations of the aforementioned methods, recent works have focused on the development of machine learning solutions that do not require inputs to be transformed into a regular time-series. These solutions can be categorised into two types, interpolation-based and non-interpolation-based models. An example of an interpolation-based model is the Discretized Multi-Time Attention (mTAND) [6] that includes a learnable interpolation component and aims to learn the temporal similarities. An example of a non-interpolation-based model is the Self-supervised Transformer

for Time-Series (STraTS) [7] which encodes each observation individually and does not depend on interpolation. While both methods are different, they managed to achieve significant success in predicting in-hospital mortality. Hence in this project, we proposed a new solution, STraTS-mTAND, that utilises both approaches to improve the in-hospital mortality prediction performance for patients admitted into the ICU. The contributions of our works are as follows:

1. Incorporated the STraTS and mTAND solutions into a single solution, STraTS-mTAND, to tackle the in-hospital mortality problem.
2. Evaluated our proposed solution, STraTS-mTAND, on two EHR datasets, PhysioNet Challenge 2012 [8] and MIMIC-III [9], and compared them to its components and other competitive baseline models for in-hospital mortality prediction.
3. Conducted experiments on different components of the PhysioNet Challenge 2012 and analysed the effects on the in-hospital mortality prediction.

This report consists of 5 sections. The first section provides a brief introduction to the project and the motivation behind this proposed solution. In section 2, we review the work done in this field. In section 3, we define the prediction problem and explain the proposed solution. Section 4 describes the experiments we conducted to test our proposed solution and evaluate the performance as well as experiments to understand the PhysioNet Challenge 2012 dataset. Lastly, section 5 concludes the report and suggests possible future works.

Chapter 2

Related Works

2.1 Irregularly sampled time-series prediction

The characteristic of irregularly sampled time-series data poses a problem to machine learning models as these models typically accept data of fixed feature size [10]. Recurrent neural networks (RNNs), which can accept time-series of different lengths, also struggle with irregularly sampled time-series data as RNNs assume that each sample in the sequence is taken at regular time intervals. Due to these limitations, new methods to utilise irregularly sampled time-series have been researched over the years to better capture the structure of irregularly sampled time-series data.

A simple approach to deal with the problems of irregular sampling is to transform the dataset into a regularly sampled dataset. Lipton et al. [11] and McDermott et al. [12] grouped the time-series into one-hour time intervals and used imputation strategies such as forward-filling and zero-imputation strategies in order to transform the irregular intervals into regular intervals. Other methods to impute the missing values include using Gaussian processes to model the time-series [13] and re-sampling methods to decide which data to impute [14]. While such an approach is simple, we have to handle situations when there are no observations within the time interval or if there is more than one observation.

An alternative approach is to construct a model that accepts the irregularly sampled

dataset as input. As mentioned in the previous section, these models can be categorised into two types, interpolation-based and non-interpolation-based models.

Interpolation-based models can be described as models that aim to generate embeddings at different timestamps. Che et al. [15] modified the inputs of the Gated Recurrent Unit (GRUs) [16] to include missing data indicators to indicate the lack of an observation and time interval since the last observation in addition to the observed values to allow the GRU to capture the irregularity of the time-series. They also proposed the Gated Recurrent Unit with trainable Decay (GRU-D) [15] that decays the input and hidden states when there are unobserved time intervals. Modifications to Long Short-Term Memory(LSTM) [17] have also been proposed. Pham et al. [18] proposed to modify the forget gate and Neil et al. [19] proposed a new time gate to regulate the access to the hidden and cell states of LSTM. For the inputs of these RNN models, the sequence length is the number of unique timestamps in the irregular time-series, which can be quite large, hence it may result in long training and inference time. To circumvent this issue, Shukla & Marlin proposed Discretized Multi-Time Attention (mTAND) [6] which uses an attention [20] mechanism instead to generate embeddings. We will elaborate on mTAND in section 2.3.

Non-interpolation-based models can be described as models that do not aim to generate representations at different timestamps instead, they view the problem of classifying the time-series as classifying a set of observations. Horn et al. [21] proposed the Set Functions for Time Series (SeFT) that take in inputs as observation triplets. It utilises differentiable set functions to summarize each observation triplet. It then passed the summarized observations to an attention mechanism to learn the importance of each observation, which is then used in the final classification layer. Self-supervised Transformer for Time-Series (STraTS) [7] proposed by Tipirneni & Reddy improves on the SeFT by using learnable embeddings. We will elaborate on STraTS in section 2.2.

2.2 Self-supervised Transformer for Time-Series (STraTS)

STraTS [7] treats the time-series as sets of observational triplets of time, feature and feature value. It considers the time and feature of an observation as the positional information and adds the time and feature embeddings to the observed value embeddings. This creates a vector for each observation that contains information on the time, feature and observed value. The observation embeddings are then passed through Transformer [20] blocks to encode the entire time-series, hence drawing dependencies between observations. Their solution resulted in an improvement of 1.0% and 0.7% in the ROC-AUC score for the MIMIC III and PhysioNet Challenge 2012 in-hospital mortality dataset respectively when compared to using SeFT.

Besides proposing a new architecture, they also utilised self-supervised learning to improve the performance on prediction task. The self-supervised learning task uses a shorter observation window from each time-series and aims to forecast the time-series variables in a short window after the observation window.

Their method does not perform any imputation or require additional missing data indicators, instead it encodes each observation individually. This method simplifies the problem by removing the irregularity and sparsity of the time-series from the input data. However, when the observations have a large time-interval, it may be difficult for the model to project the trajectory of the patients condition. Hence having interpolation can assist the model in providing some information when the observations are far apart.

2.3 Discretized Multi-Time Attention (mTAND)

Shukla and Marlin [6] proposed an attention-based mechanism that uses irregular timings where an observation was recorded to interpolate the time-series and estimate the time-series at regular predefined time points. In the attention mechanism, the irregular timings form the key, the regular predefined time points form the query and the value of the observations forms the value. The output of the attention layer is the estimated values of time-series variables at the regular predefined time points. Instead of selecting

the imputation techniques to use, the attention mechanism provides learnable parameters to understand the temporal similarity from the training data. In their experiments, their way of interpolation outperforms other imputation methods for interpolating the PhysioNet Challenge 2012 dataset. Through this learnable interpolation, we can obtain a regularly sampled, fixed-size representation of the time-series, solving the problem of irregularity and misalignment of the observations. This fixed-size representation can then be used in RNNs to perform other prediction tasks such as classification.

Their use of attention focuses on generating the embeddings of the time-series at regular timestamps to allow the classifier to learn the trajectory of the patients condition. This is drastically different from STraTS. While this method has proven successful, the interpolation may result in noise and loss of information from the irregularity of the dataset. Hence in our project, we aim to utilise both STraTS and mTAND representation of the time-series to provide the final prediction layer with a more robust representation of the time-series.

Table 2.1 summarises the different techniques developed to tackle prediction problems involving irregularly sampled time-series.

Technique	Interpolation based	Underlying Architecture Type
GP-VAE [13]	Yes	Autoencoders
GRU-D [15]	Yes	RNN
Phased LSTM [19]	Yes	RNN
SeFT [21]	No	Attention
STraTS [7]	No	Attention
mTAND [6]	Yes	Attention

Table 2.1: Summary of related works. "Interpolation based" indicates whether the technique aims to transform time-series into regular intervals. "Underlying Architecture Type" indicates the underlying neural network architecture.

Chapter 3

Proposed Solution / Methodology

As discussed in the previous section, interpolation-based methods and non-interpolation-based methods have both shown success, however both have their flaws. The interpolations generated by interpolation-based methods may not be accurate and may generate noise. Whereas, the non-interpolation-based methods use the observations alone to make predictions, which may miss the opportunity to recover information through interpolations. Hence our proposed solution takes a model from each of these two approaches and combines them to tackle the irregularly sampled time-series prediction problem. The two models provide two different ways of encoding the time-series, each approach capturing a different representation of the same time-series. Thus this allows us to generate a more robust representation of the time-series for the prediction problem. In this chapter, we first start by defining the problem and the inputs followed by the architecture of our proposed solution.

3.1 Medical Context

In our study, we focused on the irregularly sampled time-series in the medical domain. The PhysioNet Challenge 2012 [8] and MIMIC-III [9] are examples of sources of such data. Each instance in these datasets corresponds to an ICU stay and comprises demographic information, such as age and gender, as well as medical information, such as heart rate and temperature. We elaborate more on these two datasets in section 4.1.

3.2 Problem Definition

Considering a supervised learning task, where the dataset $\mathcal{D}$ contains N instances. This dataset has V time-series variables and D static variables. Time-series variables refer to variables that change with time. In the MIMIC-III dataset, examples of time-series variables are heart rate or temperature. Static variables refer to variables that stay approximately constant through the time-series. Examples of static variables in the MIMIC-III dataset are age and gender.

The dataset $\mathcal{D} = \{(\mathbf{T}_j^s, \mathbf{T}_j^m, \mathbf{d}_j, y_j) | j = 1, \ldots, N\}$ consists of four main components. The j^{th} instance in the dataset consists of the target variable y_j, static variables vector $\mathbf{d}_j \in \mathbb{R}^D$ and two representations of the time-series variables. We elaborate on the two representations in the following paragraphs. Consider an instance in the dataset where the time-series has n^s observations recording in n^t unique timestamp.

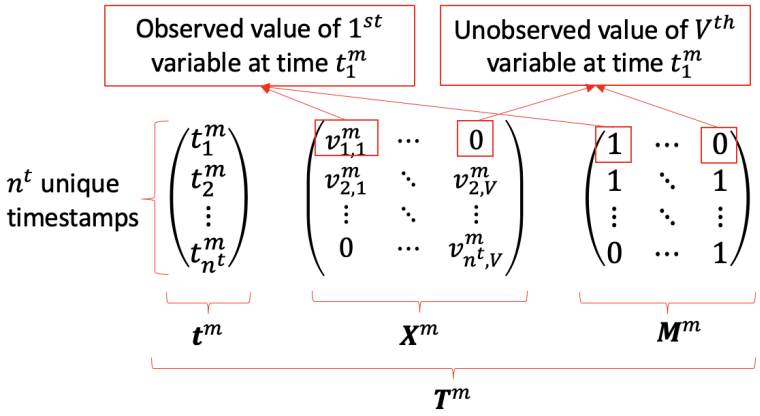

Figure 3.1: $\mathbf{T}^m$ time-series

The first time-series representation is $\mathbf{T}^m = (\mathbf{t}^m, \mathbf{X}^m, \mathbf{M}^m)$ as shown in Figure 3.1. $\mathbf{t}^m \in \mathbb{R}_{\geq 0}^{n^t}$ is the observation time vector that lists the timestamps where an observation is recorded. $\mathbf{X}^m \in \mathbb{R}^{n^t \times V}$ is the observation time matrix which stores the value of the observations. $\mathbf{M}^m \in \{0, 1\}^{n^t \times V}$ is the observation time mask that indicates 1 when an observation in $\mathbf{X}^m$ is observed and 0 when unobserved. The observation for the k^{th} variable recorded at time t_g^m, where $g \in \{1, 2, ..., n^t\}$ can be found at $\mathbf{X}_{g,k}^m$ with $\mathbf{M}_{g,k}^m = 1$. For variables that are not recorded at time t_g^m, $\mathbf{X}_{g,k}^m = 0$ with $\mathbf{M}_{g,k}^m = 0$.

The second representation of the time-series represents the time-series as sets of obser-

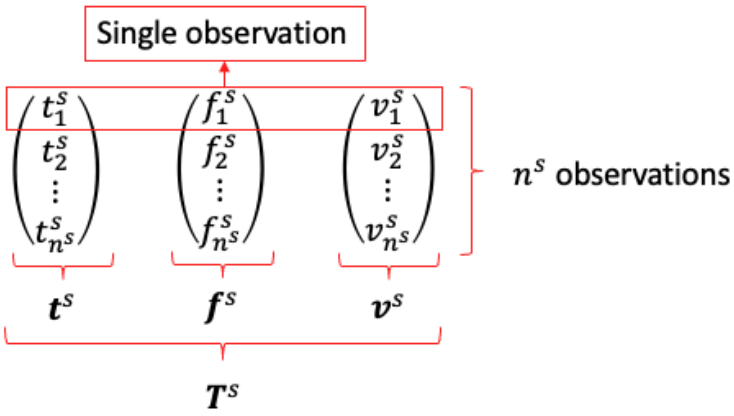

Figure 3.2: $\mathbf{T}^s$ time-series

vational triplets $\mathbf{T}^s = \{(t_k^s, f_k^s, v_k^s)\}_{k=1}^{n^s}$ as shown in Figure 3.2. Each set of observation consists of the time $t_k^s \in \mathbb{R}_{\geq 0}$, feature $f_k^s \in \mathcal{F}$ and the feature value $v_k^s \in \mathbb{R}$.

In the first representation $\mathbf{T}^m$, the time-series is represented in a value matrix $\mathbf{X}^m$, where each column represents the observed value of a variable. Due to the irregular nature of the time-series, it results in this matrix being sparse and this sparsity varies throughout the dataset. Whereas in the second representation $\mathbf{T}^s$, the time-series is represented by observations. This removes the sparsity of the time-series, resulting in a compact representation. However, due to the fixed-size observation value matrix representation $\mathbf{X}^m$ in $\mathbf{T}^m$, we do not need to perform any encoding to represent the variables as it is already represented by the column in the matrix, whereas in the second representation, we need an additional vector $\mathbf{f}^s$ to represent the variables.

3.3 Proposed Model

Our proposed model(STraTS-mTAND) modifies the STraTS, a non-interpolation-based model, by incorporating mTAND, an interpolation-based model. In this section, we refer to the STraTS module as NI-module (Non-Interpolation module) and mTAND module as the I-module (Interpolation module). Figure 3.3 shows the overview architecture of STraTS-mTAND. In this section, we will be elaborating on the individual components.

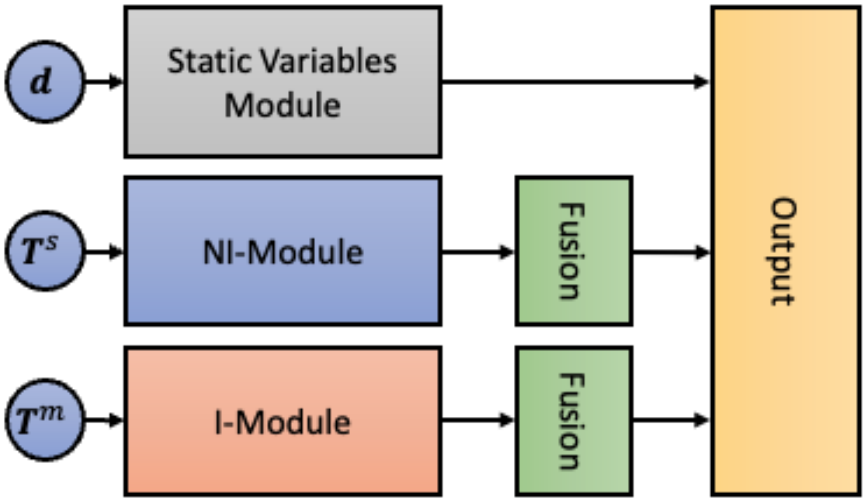

Figure 3.3: Overall Architecture of STraTS-mTAND

3.3.1 NI-module Embeddings

The NI-module takes the time-series as sets of observational triplets $\mathbf{T}^s = \{(t_k^s, f_k^s, v_k^s)\}_{k=1}^{n^s}$ Each set of observation consists of the time t_k^s, feature f_k^s and the feature value v_k^s. The output of the NI-module is the observations embedding $\mathbf{C} \in \mathbb{R}^{n^s \times d}$. Figure 3.4 shows the overview of the process used to generate the observations embeddings $\mathbf{C}$.

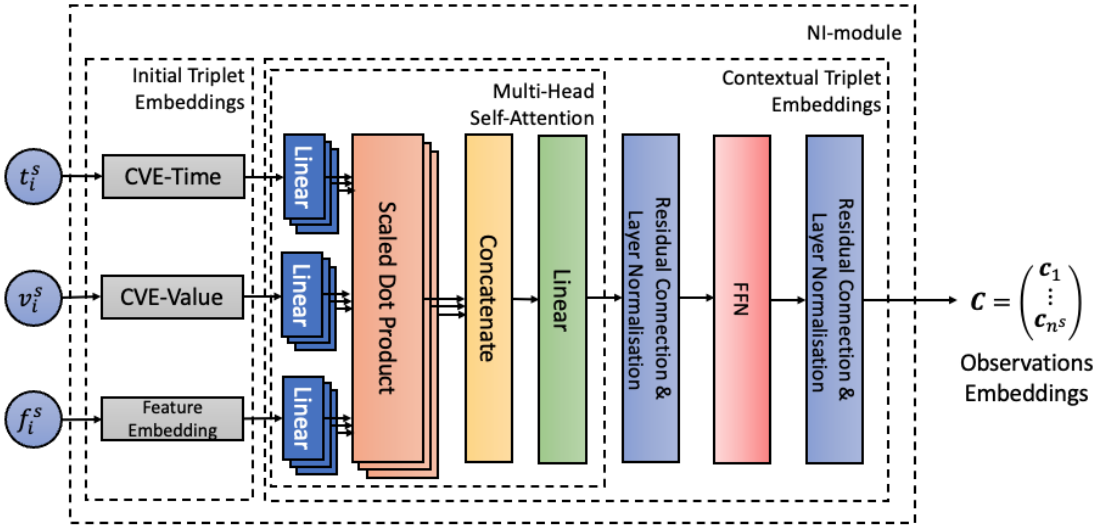

Figure 3.4: Architecture of NI-module

Initial Triplet Embedding

Each component of the k^{th} observation is first encoded individually. As the feature f_k is categorical, it is encoded using a lookup table that converts it into feature embedding $\mathbf{e}_k^f \in \mathbb{R}^d$. Time t_k^s and feature value v_k^s are continuous, hence they are encoded using the Continuous Value Embedding(CVE) module. The CVE module consists of two dense layers with the hidden layer having a dimension of $\sqrt{d}$ and an output dimension of d. The parameters of the CVE modules are $\mathbf{W}_1^{CVE} \in \mathbb{R}^{\sqrt{d}}$, $\mathbf{b}_1^{CVE} \in \mathbb{R}^{\sqrt{d}}$ and $\mathbf{W}_2^{CVE} \in \mathbb{R}^{d \times \sqrt{d}}$.

$$CVE(x) = \mathbf{W}_2^{CVE} \tanh(\mathbf{W}_1^{CVE} x + \mathbf{b}_1^{CVE}) \tag{3.1}$$

Time and feature values each have their own CVE module that encodes them into time embedding $\mathbf{e}_k^t \in \mathbb{R}^d$ and value embedding $\mathbf{e}_k^v \in \mathbb{R}^d$. The time embedding $\mathbf{e}_k^t$, feature embedding $\mathbf{e}_k^f$ and feature value embedding $\mathbf{e}_k^v$ are added together to form the initial triplet embedding $\mathbf{i}_k \in \mathbb{R}^d$ that represents the time t_k^s, feature f_k^s and feature value v_k^s of the k^{th} observation.

$$\mathbf{i}_k = \mathbf{e}_k^t + \mathbf{e}_k^f + \mathbf{e}_k^v \tag{3.2}$$

This is done for all n^s observations in the time-series, resulting in an initial triplet embeddings $\mathbf{I} = \{\mathbf{i}_1, \ldots, \mathbf{i}_{n^s}\} \in \mathbb{R}^{n^s \times d}$ for the entire time-series. What we have obtained here are embeddings for each observation that are encoded individually.

Contextual Triplet Embedding

The initial triplet embeddings $\mathbf{I}$ are then passed to the Contextual Triplet Embedding module, which consists of M Transformer [20] blocks. It outputs the contextual triplet embeddings $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_{n^s}\} \in \mathbb{R}^{n^s \times d}$. Each block consists of a Multi-Head Self-Attention layer with h heads followed by two dense layers with the hidden layer having a dimension of $2d$. The calculations for each block are as follows.

The Multi-Head Self-Attention layer takes in the input embeddings $\mathbf{I} \in \mathbb{R}^{n^s \times d}$ and

outputs self-attention embeddings $\mathbf{H} \in \mathbb{R}^{n^s \times d}$ between the observations.

$$\mathbf{H}_j = \text{softmax}\left(\frac{(\mathbf{I}\mathbf{W}_j^q)(\mathbf{I}\mathbf{W}_j^k)^T}{\sqrt{d/h}}\right)(\mathbf{I}\mathbf{W}_j^v) \quad \text{for } j = 1, \ldots, h \tag{3.3}$$

$$\mathbf{H} = (\mathbf{H}_1 \circ \cdots \circ \mathbf{H}_h)\mathbf{W}_c \tag{3.4}$$

The Multi-Head Self-Attention layer projects the input embeddings into h different queries, keys and values using $\{\mathbf{W}_j^q, \mathbf{W}_j^k, \mathbf{W}_j^v\} \in \mathbb{R}^{d \times d_h}$ respectively for each head_j. The queries and keys are used to compute the attention weight matrix. Drop Attention is applied to the attention weight matrix to randomly drop elements during training to reduce overfitting [22]. The attention weights are then used to compute the weighted averages of the embedings between n^s observations. Lastly, the outputs of each head H_j are concatenated and reduced to the original dimension d using $\mathbf{W}^c \in \mathbb{R}^{(h\dot{d}_h) \times d}$, resulting in $\mathbf{H}$. Residual connection and layer normalisation are then performed on $\mathbf{H}$ before it is passed into a two-layer feed-forward network(FFN).

$$FFN(\mathbf{X}) = \mathbf{W}_2^{FFN}\text{ReLU}(\mathbf{W}_1^{FFN}\mathbf{X} + \mathbf{b}_1^{FFN}) + \mathbf{b}_2^{FFN} \tag{3.5}$$

Dropout, residual connection and layer normalisation are applied to the output of FFN.

The above process is repeated through the M Transformer blocks with the inputs for subsequent Transformer blocks as the output of the previous, resulting in the observations embeddings $\mathbf{C}$.

In this process, the self-attention mechanism draws dependencies between the observations.

3.3.2 I-module Embeddings

For the I-module, it takes the irregularly sampled time-series as $\mathbf{T}^m = (\mathbf{t}^m, \mathbf{X}^m, \mathbf{M}^m)$. The inputs are also accompanied by a regular interval time query $\mathbf{t}^q \in \mathbb{R}^{n^q}$. The output of the I-module is the regular time embeddings $\mathbf{R} \in \mathbb{R}^{n^q \times d}$. Figure 3.5 shows the overview of the process used to generate the regular time embeddings $\mathbf{R}$.

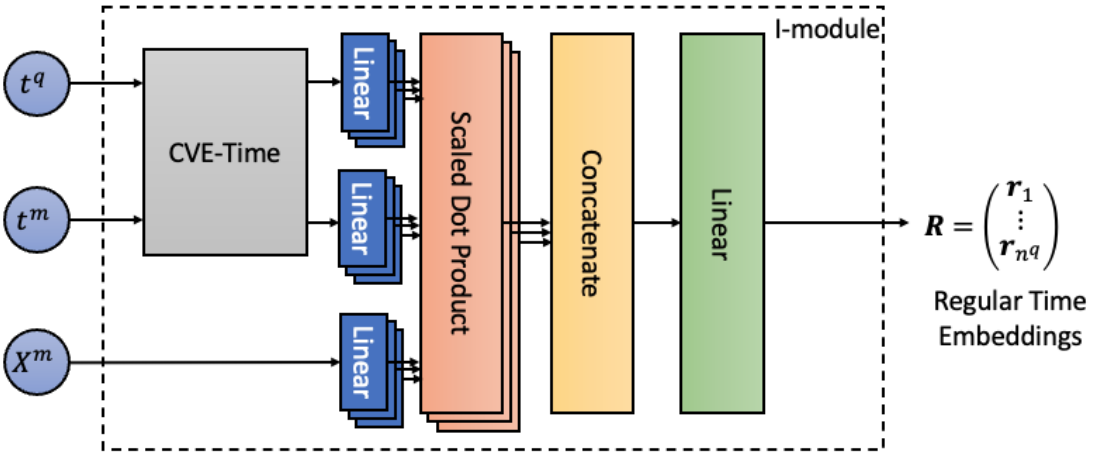

Figure 3.5: Architecture of I-module

Regular Interval Time Query

The regular interval time query $\mathbf{t}^q$ is a vector with consecutive values being separated by a constant difference. $\mathbf{t}^q$ depends on the length of the time-series problem t and the number of intervals n^q which is a hyperparameter.

The length of the time-series problem t is specific to the prediction problem and refers to the time at which all time-series end. In the case of the PhysioNet Challenge 2012, the in-hospital mortality task uses the first 48 hours of data in its prediction, hence the length of the time-series problem is 48 hours.

The number of intervals n^q is a hyperparameter and it refers to the number of time points that we want the model to interpolate using the mTAND module.

With the two parameters, the time query is defined as the following:

$$\mathbf{t}^q = \left(0, \ldots \frac{i \cdot t}{n^q}, \ldots, t\right)^T \text{ for } i \in \{1, 2, 3, ..., n^q\} \tag{3.6}$$

This time query will inform the attention mechanism in the mTAND module at which time point to produce interpolation.

Initial Time Embedding

The time inputs, $\mathbf{t}^m$ and $\mathbf{t}^q$, are first encoded using the CVE module. This CVE module uses different parameters from the two CVE modules in NI-module. This CVE module encodes $\mathbf{t}^m$ into the time key embeddings $\mathbf{E}^{t^m} \in \mathbb{R}^{n^m \times d}$ and t^q into the time query embeddings $\mathbf{E}^{t^q} \in \mathbb{R}^{n^q \times d}$.

Regular Time Embedding

After encoding the time into vectors, the inputs are passed to a Multi-Head Attention mechanism with h heads followed by a series of dense layers. It takes in three matrices as inputs, $\mathbf{E}^{t^q}$, $\mathbf{E}^{t^m}$ and $\mathbf{X}^m$, and outputs a fixed-size representation of the time series $\mathbf{R} \in \mathbb{R}^{n^q \times d}$ at regular time points defined by the time query $\mathbf{t}^q$. The calculations are as follows.

The Multi-Head Attention Layer takes in the time query embeddings $\mathbf{E}^{t^q}$ as the query, the time embeddings of the observations $\mathbf{E}^{t^m}$ as the key and the time matrix $\mathbf{X}^m$ as the value for the attention.

$$\mathbf{H}_j = \text{softmax}\left(\frac{(\mathbf{t}^q\mathbf{W}_j^q)(\mathbf{t}^m\mathbf{W}_j^k)^T}{\sqrt{d/h}}\right)(\mathbf{X}^m\mathbf{W}_j^v) \text{ for } j = 1, \ldots, h \tag{3.7}$$

$$\mathbf{H} = (\mathbf{H}_1 \circ \cdots \circ \mathbf{H}_h) \tag{3.8}$$

Similar to the Multi-Head Self-Attention layer, the query, key and value get projected into h different subspace using $\mathbf{W}_j^q \in \mathbb{R}^{d \times d_h}$, $\mathbf{W}_j^k \in \mathbb{R}^{d \times d_h}$, $\mathbf{W}_j^v \in \mathbb{R}^{V \times V}$ respectively for each head j. Each head j will give us a different representation of the same time series $\mathbf{H}_j \in \mathbb{R}^{n^q \times V}$ and concatenating them along the last axis results in a three-dimension matrix $\mathbf{H} \in \mathbb{R}^{n^q \times V \times h}$. This three-dimensional matrix is then passed through a series of dense layers. The parameters for the series of dense layers are $\mathbf{W}_1^m \in \mathbb{R}^{h \times d}$, $\mathbf{b}_1^m \in \mathbb{R}^d$, $\mathbf{W}_2^m \in \mathbb{R}^{\sqrt{V} \times V}$, $\mathbf{b}_2^m \in \mathbb{R}^{\sqrt{V}}$, $\mathbf{W}_3^m \in \mathbb{R}^{1 \times \sqrt{V}}$ and $\mathbf{b}_3^m \in \mathbb{R}^d$

$$\mathbf{r}_k^T = \mathbf{W}_3^m(\tanh(\mathbf{W}_2^m\tanh(\mathbf{H}_{k,*,*}\mathbf{W}_1^m + \mathbf{b}_1^m) + \mathbf{b}_2^m)) + \mathbf{b}_3^m \tag{3.9}$$

$$\mathbf{R} = (\mathbf{r}_1 \circ \cdots \circ \mathbf{r}_{t^q}) \tag{3.10}$$

In the dense layers, we fuse the different representations $\mathbf{H}_{k,*,*}$ at time $\mathbf{t}_k^q$ to obtain a fixed dimension embedding $\mathbf{r}_k \in \mathbb{R}^d$ for the specified time $\mathbf{t}_k^q$. This is done for all time points and concatenated to obtain the regular time embeddings $\mathbf{R}$.

In this process, the I-module utilises the attention mechanism to produce h representations of the interpolated time-series, which is subsequently used to generate the regular time embeddings $\mathbf{R}$ through a series of dense layers.

3.3.3 Fusion Self-Attention

The observations embeddings $\mathbf{C}$ and regular time embeddings $\mathbf{R}$ each get passed into two different Fusion Self-Attention layers (FSA). The Fusion Self-Attention layer is used to generate the attention weights α for the embeddings to fuse them into a vector that represents the information needed in the prediction problem. It consists of two dense layers with parameters $\mathbf{W}_1^{FSA} \in \mathbb{R}^{d\times 2d}$, $\mathbf{b}_1^{FSA} \in \mathbb{R}^{2d}$, $\mathbf{W}_2^{FSA} \in \mathbb{R}^{2d\times 1}$ and performs the following calculations.

$$FSA(\mathbf{E}) = \text{softmax}\left(\tanh\left(\mathbf{E}\mathbf{W}_1^{FSA} + \mathbf{b}_1^{FSA}\right)\mathbf{W}_2^{FSA}\right) \tag{3.11}$$

For the observations embeddings, the output is $FSA(\mathbf{C}) = \boldsymbol{\alpha}^s \in \mathbb{R}^{n^s}$. For the regular time embeddings, the output is $FSA(\mathbf{R}) = \boldsymbol{\alpha}^m \in \mathbb{R}^{n^q}$. The embeddings are then multiplied with the respective attention weights α to generate the NI-embeddings $\mathbf{e}^s \in \mathbb{R}^d$ and I-embeddings $\mathbf{e}^m \in \mathbb{R}^d$.

$$\mathbf{e}^s = \mathbf{C}^T\boldsymbol{\alpha}^s \tag{3.12}$$

$$\mathbf{e}^m = \mathbf{R}^T\boldsymbol{\alpha}^m \tag{3.13}$$

3.3.4 Static Variable Embeddings

Since the static variables $\mathbf{d} \in \mathbb{R}^D$ do not change with time, it is encoded separately from the time-series variables through two dense layers with parameters $\mathbf{W}_1^D \in \mathbb{R}^{1\times 2d}$, $\mathbf{b}_1^D \in \mathbb{R}^{2d}$, $\mathbf{W}_2^D \in \mathbb{R}^{2d\times d}$, $\mathbf{b}_2^D \in \mathbb{R}^d$. This provides us with the static variable embeddings

$\mathbf{e}^D \in \mathbb{R}^d$.

$$\mathbf{e}^D = \tanh\left(\tanh\left(\mathbf{d}\mathbf{W}_1^D + \mathbf{b}_1^D\right)\mathbf{W}_2^D\right) + \mathbf{b}_2^D \tag{3.14}$$

3.3.5 Output Layer

The two sets of time embeddings, $\mathbf{e}^m$ and $\mathbf{e}^s$, and the static variable embeddings $\mathbf{e}^D$ are then concatenated and fed through two dense layers with weights $\mathbf{W}_1^O \in \mathbb{R}^{3d\times V}$, $b_1^O \in \mathbb{R}^V$ and $\mathbf{W}_2^O \in \mathbb{R}^{V\times 1}$, $b_2^O \in \mathbb{R}$, followed by a sigmoid activation function to produce the final output $\hat{y}$.

$$\hat{y} = \text{sigmoid}\left(\left([\mathbf{e}^m \circ \mathbf{e}^s \circ \mathbf{e}^D]\mathbf{W}_1^O + b_1^O\right)\mathbf{W}_2^O + b_2^O\right) \tag{3.15}$$

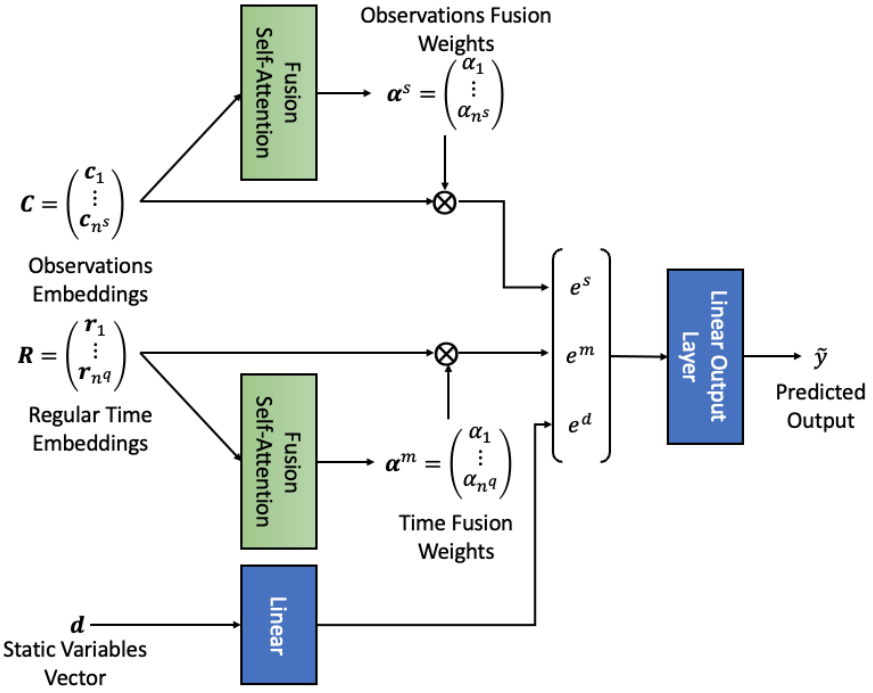

Figure 3.6: Architecture of Static Variable, Fusion and Output module

All in all, our proposed solution STraTS-mTAND utilised both interpolation-based mTAND and non-interpolation-based STraTS to generate two different representations of the time-series, $\mathbf{e}^s$ and $\mathbf{e}^m$ and use them in the prediction task. Table 3.1 provides a summary of the mathematical notations mentioned in this section.

Notation	Definition
$N \in \mathbb{N}$	# of time-series in dataset
$V \in \mathbb{N}$	# of time-series variables
$D \in \mathbb{N}$	# of static variables
$\mathcal{F}$	Set of V time-series variables
$\mathbf{d} \in \mathbb{R}^D$	Vector for D static variable
$n^s \in \mathbb{N}$	# of observations in the time-series
$t_k^s \in \mathbb{R}_{\leq 0}$	Time of k^{th} observation
$f_k^s \in \mathcal{F}$	Feature of k^{th} observation
$v_k^s \in \mathbb{R}$	Value of k^{th} observation
$\mathbf{T}^s = \{(t_k^s, f_k^s, v_k^s)\}_{k=1}^{n^s}$	Time-series representation for NI-module
$\mathbf{I} = (\mathbf{i}_1, \ldots, \mathbf{i}_{n^s})^T$	Initial Triplet Embeddings for the time-series
$\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_{n^s})^T$	Observations Embeddings
$n^m \in \mathbb{N}$	# of timestamps with at least 1 observation
$t^m \in \mathbb{R}^{n^m}$	Observation Time Vector
$\mathbf{X}^m \in \mathbb{R}^{n^m \times V}$	Observation Time Matrix that records values of observations
$\mathbf{M}^m \in \mathbb{R}^{n^m \times V}$	Observation Time Mask that indicates 1 when a variable is observed and 0 otherwise
$\mathbf{T}^m = \{\mathbf{t}^m, \mathbf{X}^m, \mathbf{M}^m\}$	Time-series representation for I-module
$n^q \in \mathbb{N}$	Length of the time query (Hyperparameter)
$\mathbf{t}^q \in \mathbb{R}^{n^q}$	Time query for I-module
$\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_{n^q})^T$	Regular Time Embeddings
$\mathbf{e}^s \in \mathbb{R}^d$	NI-Embeddings
$\mathbf{e}^m \in \mathbb{R}^d$	I-Embeddings
$\mathbf{e}^D \in \mathbb{R}^d$	Static Variable Embeddings
$\hat{y} \in [0, 1]$	Output for for binary classification task

Table 3.1: Mathematical Notations

Chapter 4

Experiments

We conducted several experiments and evaluated our proposed model against other models using the PhysioNet Challenge 2012 and MIMIC-III dataset for the in-hospital mortality prediction task. In this section, we present the dataset details, implementation details and the results of the experiments we performed.

4.1 Dataset

PhysioNet Challenge 2012

The PhysioNet Challenge 2012 dataset was curated from the MIMIC-II Clinical Database, version 2.6 [23]. The MIMIC-II dataset consists of 25,328 ICU stays which occurred at Beth Israel Deaconess Medical Center between 2001 to 2007. For the PhysioNet Challenge 2012 dataset, the organisers filtered for ICU stays that last for at least 48 hours with patients aged larger than 16 years old on admission and randomly selected 12,000 of such stays. There are a total of 37 time-series variables and 4 static variables. The dataset has multiple prediction problems, such as in-hospital mortality and length of stay. For our project, we will be focusing on the in-hospital mortality prediction Following the original problem statement in the PhysioNet Challenge, the prediction problem is to predict the in-hospital mortality using the first 48 hours of data. The PhysioNet Challenge 2012 data is split into 3 different sets, A, B and C.

Following the STraTS paper, sets B and C form the training and validation dataset in an 80:20 split and set A forms the test dataset. Table 4.1 shows the general descriptions of the dataset.

	Training	Validation	Test	All
# of ICU Stays	6,392	1,599	3,997	11,988
Avg. span of time-series(hours)	47.3	47.4	47.2	47.3
Avg. # observations/stay	435.7	437.5	434.7	435.6
Max # observations/stay	1191	1156	1497	1497
% of positive labels	14.1	15.6	13.9	14.2

Table 4.1: General characteristic of PhysioNet Challenge 2012 dataset

The 4 static variables are general descriptors of the patient that were collected upon admission. The 4 variables collected are age, gender, height and the type of ICU the patient was admitted to. Figure 4.1 shows the distribution of the static variables. For the ICU Type variable, we performed one-hot encoding, producing four new columns.

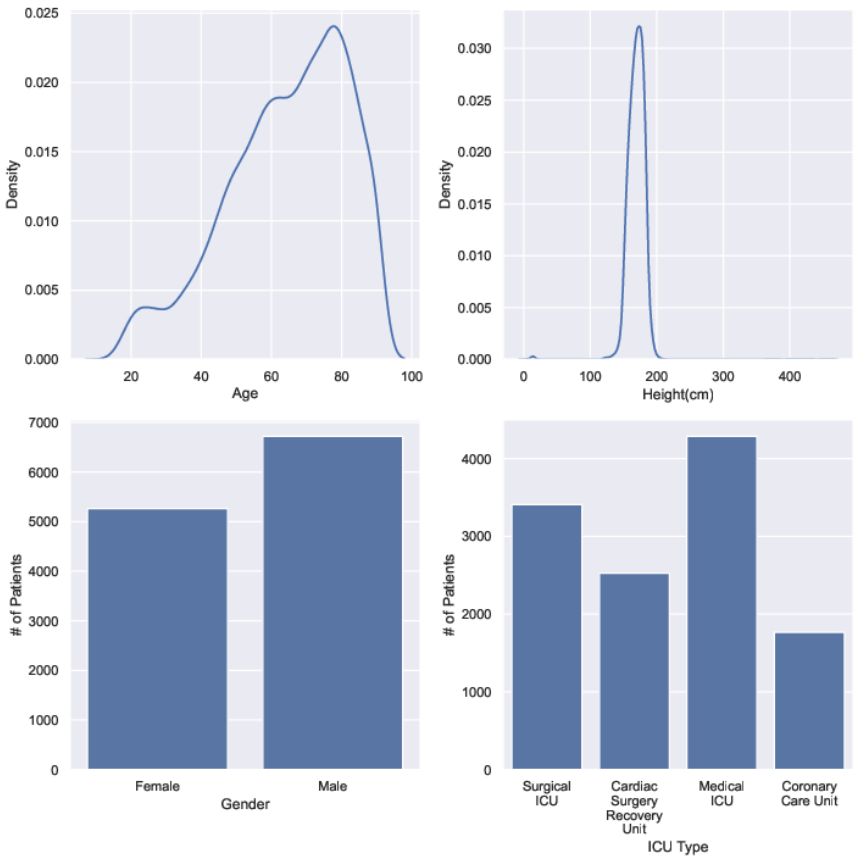

Figure 4.1: Distribution of Static Variables

As mentioned, there are 37 time-series variables in this dataset and they can be collected

at any given time. Each observation has a timestamp that records the timing of the observation since the patient admission into the ICU. Table 4.2 shows the characteristics of the time-series variables for all ICU stays. We can see that there are variables such as Cholesterol appearing only in 7.91% of all stays and White Blood Cell Count appearing on average 3.25 times in 48 hours. This highlights the sparsity and irregularity of the data.

From Figure 4.2, the distribution of the number of total observations shows that it is slightly right-skewed, with the top 10% ranging from 596 to 1497 observations. 97% of patients have at least 1 observation within the first hour of admission. This percentage gradually decreases with time to 91.7% in the 23rd hour.

From Figure 4.3, the number of hourly observations has generally decreased with time. On further analysis of the trend of hourly observations per patient across time, the correlation plot shows that there is close to no correlation across time. This points out the irregularity of the number of observations across time in the dataset.

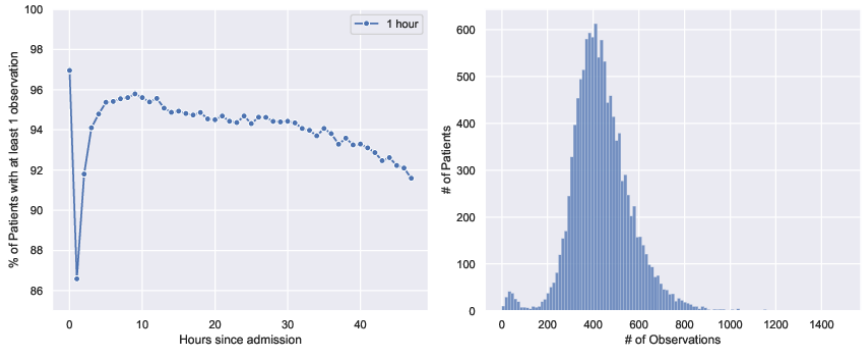

Figure 4.2: Distribution of Observations

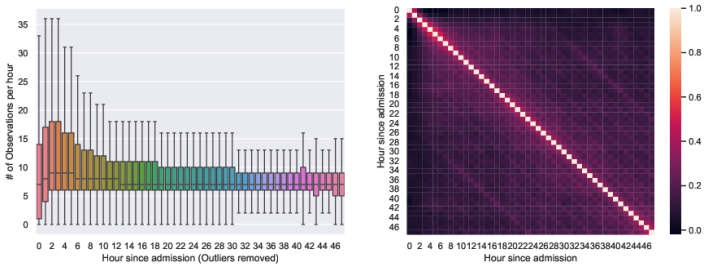

Figure 4.3: Number of Observations

Time-series Variable	Average # of Observations per patient	% of patients with at least one observation
ALP	0.79	42.52
ALT	0.81	43.49
AST	0.81	43.49
Albumin	0.60	40.63
BUN	3.48	98.57
Bilirubin	0.82	43.49
Cholesterol	0.08	7.91
Creatinine	3.50	98.57
DiasABP	36.71	70.29
FiO2	7.94	67.70
GCS	15.51	98.56
Glucose	3.28	97.65
HCO3	3.41	98.37
HCT	4.59	98.52
HR	57.32	98.57
K	3.65	98.02
Lactate	2.03	54.85
MAP	36.78	70.14
MechVent	7.64	63.24
Mg	3.41	97.66
NIDiasABP	24.58	87.33
NIMAP	24.23	87.29
NISysABP	24.61	87.69
Na	3.42	98.33
PaCO2	5.76	75.53
PaO2	5.75	75.53
Platelets	3.56	98.45
RespRate	13.80	27.76
SaO2	2.00	44.78
SysABP	36.72	70.30
Temp	21.64	98.56
TroponinI	0.10	4.71
TroponinT	0.52	22.00
Urine	34.30	97.51
WBC	3.25	98.32
Weight	32.19	92.43
pH	6.02	75.98

Table 4.2: Missingness of features in PhysioNet-2012 dataset

MIMIC-III

The MIMIC-III dataset contains medical records of 46,476 patients admitted to the ICU of Beth Israel Deaconess Medical Center between 2001 and 2012 [24]. To extract and

clean the dataset, we utilised the implementation by Harutyunyan et al. [24]. In their implementation, they selected 17 time-series variables that represent a subset of the 37 variables from the PhysioNet Challenge 2012. Using the same logic, since only age and gender can be found in the MIMIC-III dataset, only 2 static variables are used. They also removed patients below the age of 18 as there is a difference between adult and pediatric physiology [24]. Following the implementation in the original STraTS paper, we define the prediction problem as predicting the in-hospital mortality for patients with ICU stays of at least one day. For the splitting of the training, validation and test sets, we utilised Harutyunyan et al. [24] predefined test set. The remaining ICU stays forms the training and validation sets in an 80:20 split. Table 4.3 shows the general description of the dataset.

	Training	Validation	Test	All
# of ICU Stays	24490	5356	5282	35128
Avg. span of time-series(hours)	23.4	23.4	23.4	23.4
Avg. # observations/stay	234	233	235	234
Max # observations/stay	9365	8859	9533	9533
% of positive labels	10.4	10.9	9.50	10.4

Table 4.3: General characteristic of MIMIC-III dataset

Figure 4.4 shows the distribution of the static variables. The age and gender distribution are relatively similar in both datasets.

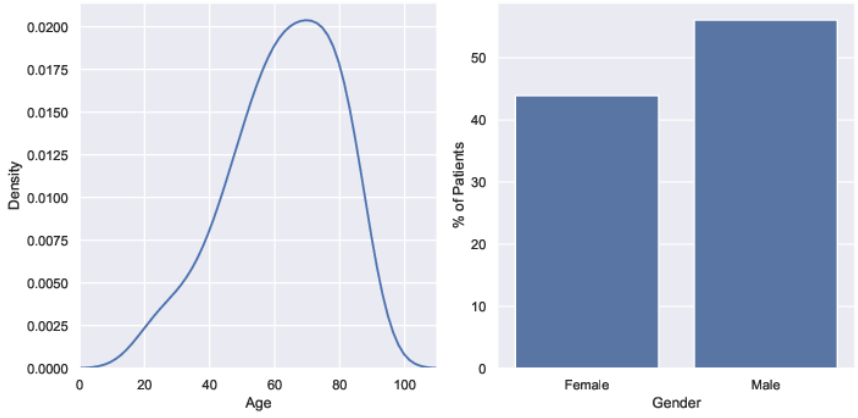

Figure 4.4: Distribution of Static Variables

Similar to the PhysioNet Challenge 2012 dataset, each observation has a timestamp that records the timing of the observation since the patient admission into the ICU. Table

4.4 shows the characteristics of the 17 time-series variables.

Time-series Variable	Average # of Observations per patient	% of patients with at least one observation
Capillary refill rate	0.08	1.23
Diastolic blood pressure	29.30	98.80
Fraction inspired oxygen	2.15	25.00
Glascow coma scale eye opening	7.81	99.05
Glascow coma scale motor response	7.77	99.02
Glascow coma scale total	4.41	55.21
Glascow coma scale verbal response	7.78	99.01
Glucose	7.71	99.22
Heart Rate	31.08	98.80
Height	0.18	17.48
Mean blood pressure	29.10	98.79
Oxygen saturation	31.44	99.23
Respiratory rate	31.31	98.70
Systolic blood pressure	29.31	98.80
Temperature	9.80	97.27
Weight	1.31	67.81
pH	3.93	69.84

Table 4.4: Missingness of features in MIMIC-III dataset

From Figure 4.5, the boxplot of the number of observations in each stay shows that the distribution is extremely right-skewed, with the top 1% ranging from 462 to 9533 observations. About 95% of the patients have at least one observation per hour most of the time. This is higher as compared to the PhysioNet Challenge 2012 dataset.

From Figure 4.6, the distribution of the number of hourly observations seems constant throughout time. On further analysis of the trend of hourly observation per patient, we plotted a correlation plot of the number of observations between different hours since admission, without ICU stays with more than 462 observations. Similar to the previous dataset, there is little to no correlation across time, thus showing the irregularity of the number of observations across time in the dataset.

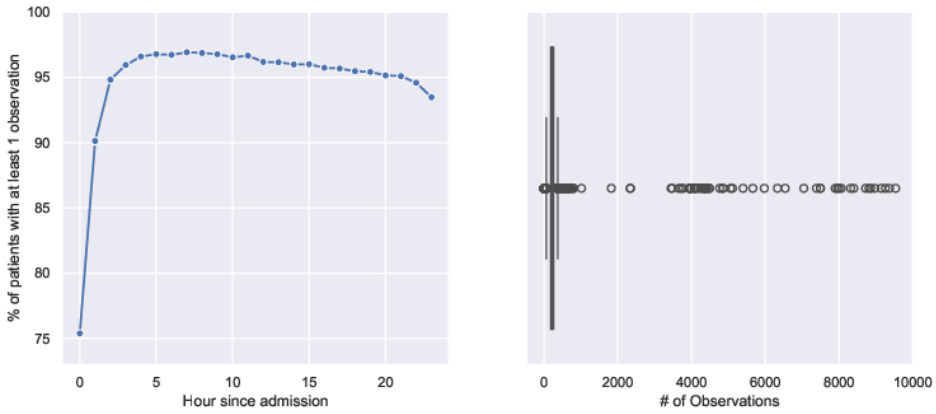

Figure 4.5: Distribution of Observations

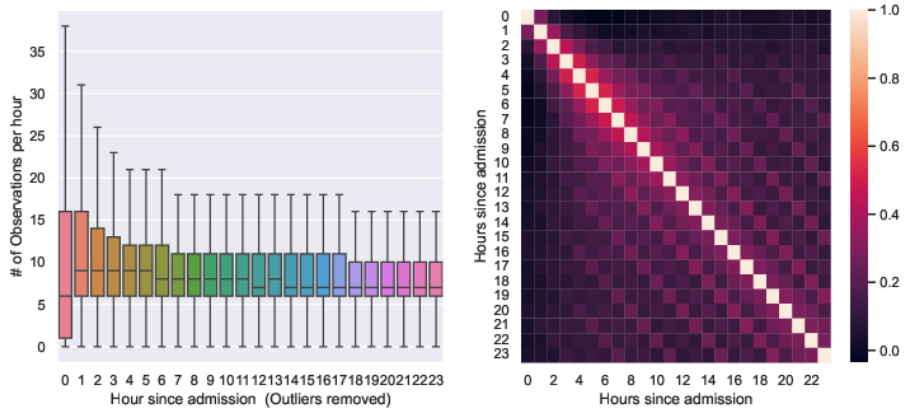

Figure 4.6: Number of Observations

4.2 Baseline Models

We compared the performance of STraTS-mTAND against several deep-learning models.

1. GRU with trainable Decay (GRU-D) [15]
2. Set Functions for Time Series (SeFT) [21]
3. Discretized Multi-Time Attention (mTAND) [6]
4. Self-supervised Transformer for Time Series (STraTS) [7]

4.3 Implementation

The time-series variables and static variables for both datasets are normalized to have zero mean and unit variance.

All the models are implemented using Keras with TensorFlow backend. For STraTS, we set the maximum number of observations using the 99^{th} percentile. This is done to avoid any memory overflow during batch gradient descent. For mTAND, we adapted the PyTorch implementation from Shukla & Marlin (GitHub) and implemented our version using Keras with Tensorflow backend. Following the implementation of the paper, the observation timestamps rounded to the nearest minute before constructing the time-series matrix. We borrowed the implementation of GRU-D and SeFT from Horn et al. (GitHub) and STraTS from Tipirneni & Reddy (GitHub). For our proposed solution, we extended the STraTS implementation provided by Tipirneni & Reddy in the above GitHub link and integrated mTAND into it.

All models were trained with a batch size of 32 using the Adam optimizer [25]. To combat the imbalance class problem, we used a weighted binary cross entropy loss function to provide higher weightage to the positive samples [26]. The training stops when the sum of ROC-AUC and PR-AUC on the validation data does not improve for 10 epochs and the model was restored using the weights with the best validation sum of ROC-AUC and PR-AUC before predicting on the test dataset. The hyperparameters used in the experiments are listed in Table 4.5.

4.4 Evaluation

As the dataset is imbalanced, we used ROC-AUC and PR-AUC to evaluate the performance of the models.

ROC-AUC

ROC-AUC refers to the area under the Receiver Operating Characteristic (ROC) curve. This curve is constructed by considering all classification thresholds and plotting the

Model	PhysioNet Challenge 2012	MIMIC-III
GRU-D	n_units=49, recurrent_dropout=0.2, dropout=0.2, lr=0.0004	n_units=60, recurrent_dropout=0.2, dropout=0.2, lr=0.0004
SeFT	lr=0.00081, n_phi_layers=4, phi_width=128, phi_dropout=0.2, n_psi_layers=2, psi_width=64, psi_latent_width=128, dot_prod_dim=128, n_heads=4, attn_dropout=0.5, latent_width=32, n_rho_layers=2, rho_width=512, rho_dropout=0.0, max_timescale=100.0, n_positional_dims=4	lr=0.001, n_phi_layers=4, phi_width=128, phi_dropout=0.2, n_psi_layers=2, psi_width=64, psi_latent_width=128, dot_prod_dim=128, n_heads=4, attn_dropout=0.5, latent_width=32, n_rho_layers=2, rho_width=512, rho_dropout=0.0, max_timescale=100.0, n_positional_dims=4
mTAND	d=32, he=8, dropout=0.2, len_time_query=200, lr=0.0005	d=32, he=8, dropout=0.2, len_time_query=100, lr=0.0005
STraTS	d=32, N=2, he=4, dropout=0.2, lr=0.0005	d=32, N=2, he=4, dropout=0.2, lr=0.0005
STraTS- mTAND	d_strats=32, N_strats=2, he_strats=4, dropout_strats=0.2, d_mtand=32, he_mtand=8, dropout_mtand=0.2, len_time_query=200, lr=0.0005	d_strats=32, N_strats=2, he_strats=4, dropout_strats=0.2, d_mtand=32, he_mtand=8, dropout_mtand=0.2, len_time_query=100, lr=0.0005

Table 4.5: Hyperparameters of Baseline Models

true positive rate against the false positive rate at all thresholds. This metric measures the ability of the model to differentiate between the two classes and serves as a combined measure of sensitivity and specificity [27]. In the context of predicting in-hospital mortality, it is essential to maximize sensitivity to capture all potential positive cases, ensuring appropriate allocation of medical resources to critical patients. However, it is equally crucial to maintain specificity to prevent over-prediction of positive cases, which could lead to inefficient resource allocation. Hence a balance between both is required. The higher the ROC-AUC, the better the performance of the model.

$$\text{ROC-AUC} = \int_{x=0}^{x=1} \text{TPR}(\text{FPR}(x))d\text{FPR}, \quad \text{where } x \text{ is the classification threshold} \tag{4.1}$$

PR-AUC

PR-AUC refers to the area under the Precision-Recall curve. Similar to the ROC-AUC, this metric takes into account all classification thresholds and plots the precision against the recall at all thresholds. As this metric uses precision and recall in the underlying calculations, it focuses on the performance of the model on the minority class [28, 29]. Thus this metric is suitable for this prediction problem where predicting the minority class, in-hospital mortality, is more important. The higher the PR-AUC, the better the performance of the model.

$$\text{PR-AUC} = \int_{x=0}^{x=1} \text{Precision}(\text{Recall}(x))d\text{Recall}, \quad \text{where } x \text{ is the classification threshold} \tag{4.2}$$

4.5 Model Related Experiments

In this section, we present the performance of our proposed model against the baseline models. The focus of the experiments in this section is on testing the models under different situations for robustness as well as justifying the addition of the mTAND module in STraTS.

4.5.1 Overall Comparison between Models

We trained the models using all training data for 10 runs and obtained the following results. The results are shown in Table 4.6 with **best** performance in **bold**. For the Physionet Challenge 2012 dataset, the STraTS-mTAND outperformed all the baseline models by at least 1% on both the ROC-AUC and PR-AUC metrics. For the MIMIC-III dataset, the average performance on the GRU-D and SeFT are comparable to the

STraTS-mTAND, with the ROC-AUC score differing by 0.1% and the PR-AUC score differing by 0.5%. However, the STraTS-mTAND outperformed STraTS in ROC-AUC marginally and 1.5% in PR-AUC. This shows that the inclusion of the mTAND module into the STraTS model provides additional information for the final classification layer. A hypothesis on why the GRU-D and SeFT performed equivalently or better than STraTS-mTAND on the MIMIC-III as compared to the PhysioNet dataset could be because of the reduced number of time-series variables, from 34 to 17.

Dataset	Model	ROC-AUC	PR-AUC
PhysioNet Challenge 2012	GRU-D	0.846 ± 0.001	0.497 ± 0.009
	SeFT	0.846 ± 0.004	0.506 ± 0.010
	mTAND	0.844 ± 0.001	0.499 ± 0.004
	STraTS	0.850 ± 0.003	0.510 ± 0.007
	STraTS-mTAND	**0.860 ± 0.004**	**0.520 ± 0.014**
MIMIC-III	GRU-D	0.870 ± 0.002	0.494 ± 0.006
	SeFT	0.870 ± 0.002	**0.499 ± 0.006**
	mTAND	0.855 ± 0.004	0.452 ± 0.018
	STraTS	0.868 ± 0.003	0.478 ± 0.010
	STraTS-mTAND	**0.871 ± 0.002**	0.493 ± 0.014

Table 4.6: Mortality prediction performance dataset averaged over 10 runs

4.5.2 Generalisation Ability of Models

We tested the generalisation ability of our proposed model to ensure that our additional mTAND module did not result in a worse generalisation of the STraTS model as shown by Tipirneni & Reddy [7] in the original paper. We experimented by randomly sampling p% from the training and validation dataset without replacement and trained the models using the sampled data. The entire test dataset was used during evaluation to ensure that the results across different p values were comparable. This process was repeated 10 times for each p value and both datasets. Figure 4.7 and Figure 4.8 show the results for the PhysioNet Challenge 2012 and MIMIC-III dataset across different p values respectively.

There is an overall decline in the performance across all models, as the models are trained on less data. However, STraTS-mTAND still generally outperforms all the models for all values of p. Hence, the addition of the mTAND module not only

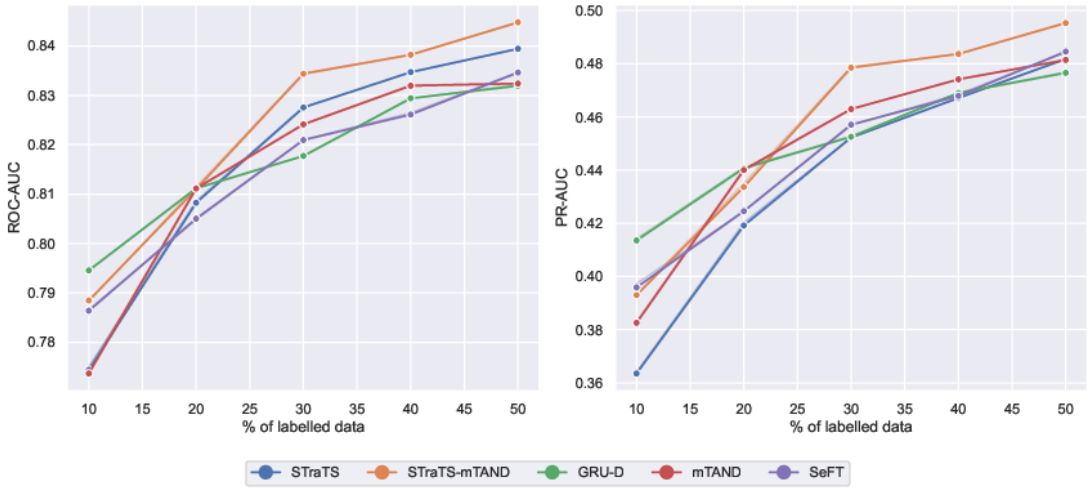

Figure 4.7: Average mortality prediction performance using different percentages of labelled data over 10 runs for PhysioNet Challenge 2012

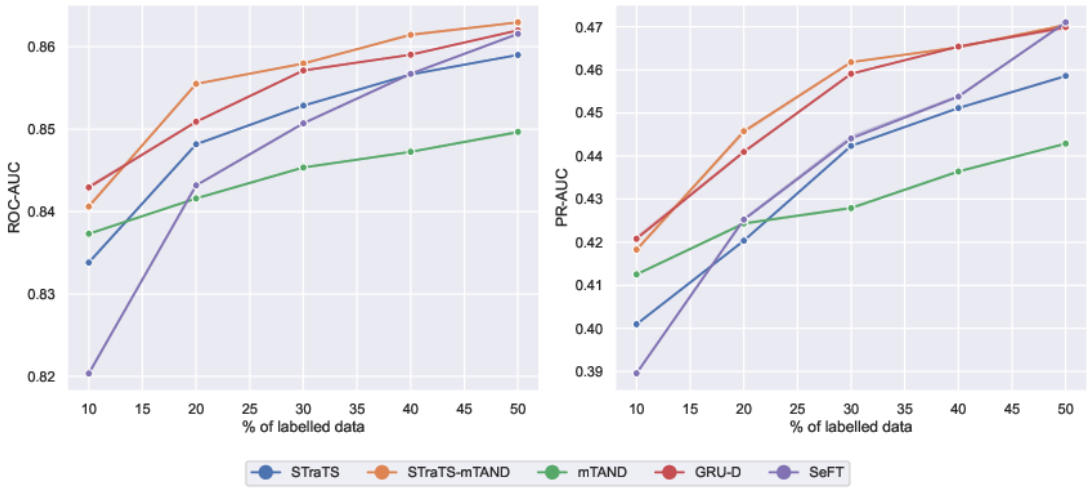

Figure 4.8: Average mortality prediction performance using different percentages of labelled data over 10 runs for MIMIC-III

improved the performance of STraTS, it also did not sacrifice the generalisation ability of STraTS.

4.5.3 Self-Supervision Training

In the original STraTS paper, the authors proposed self-supervised forecasting training before training the model on classification to improve the performance of mortality prediction.

Following the forecasting task proposed in the original STraTS paper, we train the model to predict the values of the time-series variable after an observation window. For the PhysioNet Challenge 2012, the observation window is defined as $\{[0, x)|12 \leq x \leq 44, x\%4 = 0\}$, while for the MIMIC-III dataset, it is defined as $\{[0, x)|12 \leq x \leq 20, x\%4 = 0\}$. The target for each time-series variable is set as the first observation within 2 hours after the observation window. Variables that are not observed within 2 hours are masked out during the computation of the loss function.

We tested the impact of self-supervision training on STraTS-mTAND and STraTS. Table 4.7 shows the results of the mortality prediction performance averaged over 10 runs with the best performance in **bold**.

Dataset	Model	Self-Supervision	ROC-AUC	PR-AUC
PhysioNet Challenge 2012	STraTS-mTAND	✓	0.855 ± 0.004	0.511 ± 0.007
		×	**0.860 ± 0.004**	**0.520 ± 0.014**
	STraTS	✓	0.855 ± 0.003	0.504 ± 0.006
		×	0.850 ± 0.003	0.510 ± 0.007

Table 4.7: Mortality prediction performance with and without self-supervision training

Unlike the STraTS model, the STraTS-mTAND does not seem to benefit from the self-supervision training. A hypothesis on why this may be the case could be due to the length of the time query n^q. In our implementation of the time query t^q, the time interval between each time point is defined as $t^q_{i+1} - t^q_i = \frac{\text{Length of Observation Window (Hours)}}{n^q}$. This time interval is constant in the mortality prediction dataset with the observation window being 24 or 48 hours. However, in the self-supervised forecasting dataset, the observation window varies across instances and ranges from 12 to 44 hours, thus the time interval also varies. This creates more variability for the model to learn. Further work can be done on testing different variations of the time query such as having time queries of different lengths for each instance or having time queries of different time intervals.

4.5.4 Impact of Time-series Sparsity

As the mTAND module performs interpolation on the input data, we expect the STraTS-mTAND to be able to hold its performance when trained on a sparser dataset. To test the robustness of STraTS-mTAND, we randomly removed timestamps and the associated observations iteratively such that only p proportion of timestamps are left in each time-series to create a sparser time-series compared to the original dataset. The above manipulations were done on training, validation and test datasets. Table 4.8 shows the aggregated statistics of the datasets after performing the above manipulations. $p = 1.0$ refers to the original dataset.

Dataset	p	Avg time between observations(Hours)	Median # of observations	Median # of timestamps
PhysioNet Challenge 2012	1.0	0.638	424	72
	0.8	0.800	337	57
	0.6	1.062	251	42
	0.5	1.227	207	35
	0.4	1.596	163	28

Table 4.8: Statistics of manipulated dataset

The results in Table 4.9 and Figure 4.9 show the average results over 10 runs. For all values of p, the STraTS-mTAND outperforms STraTS across all variables. We also observed that for the PR-AUC metric, the mTAND model can hold its performance better as compared to STraTS, as it has a similar PR-AUC score for $p \in \{0.6, 0.4\}$ despite having a lower PR-AUC score at $p = 1.0$. Hence this suggests that the interpolations help to estimate the patients condition during long intervals without any observations and provide the classification layer with useful information for the prediction problem.

4.6 PhysioNet Challenge 2012 Related Experiments

Besides testing our model, we also conducted two experiments to understand how the different components of the PhysioNet Challenge 2012 dataset contribute to the performance of the STraTS-mTAND.

p	Model	ROC-AUC	PR-AUC
1.0	STraTS	0.850 ± 0.003	0.510 ± 0.007
	mTAND	0.844 ± 0.001	0.499 ± 0.004
	STraTS-mTAND	0.860 ± 0.004	0.520 ± 0.014
0.8	STraTS	0.853 ± 0.003	0.507 ± 0.006
	mTAND	0.846 ± 0.002	0.503 ± 0.003
	STraTS-mTAND	0.860 ± 0.004	0.519 ± 0.014
0.6	STraTS	0.843 ± 0.005	0.489 ± 0.017
	mTAND	0.839 ± 0.002	0.489 ± 0.007
	STraTS-mTAND	0.853 ± 0.003	0.515 ± 0.008
0.5	STraTS	0.837 ± 0.004	0.492 ± 0.011
	mTAND	0.830 ± 0.002	0.479 ± 0.003
	STraTS-mTAND	0.844 ± 0.003	0.499 ± 0.011
0.4	STraTS	0.836 ± 0.006	0.473 ± 0.015
	mTAND	0.826 ± 0.004	0.494 ± 0.009
	STraTS-mTAND	0.841 ± 0.004	0.472 ± 0.004

Table 4.9: Mortality prediction performance with datasets of different sparsity

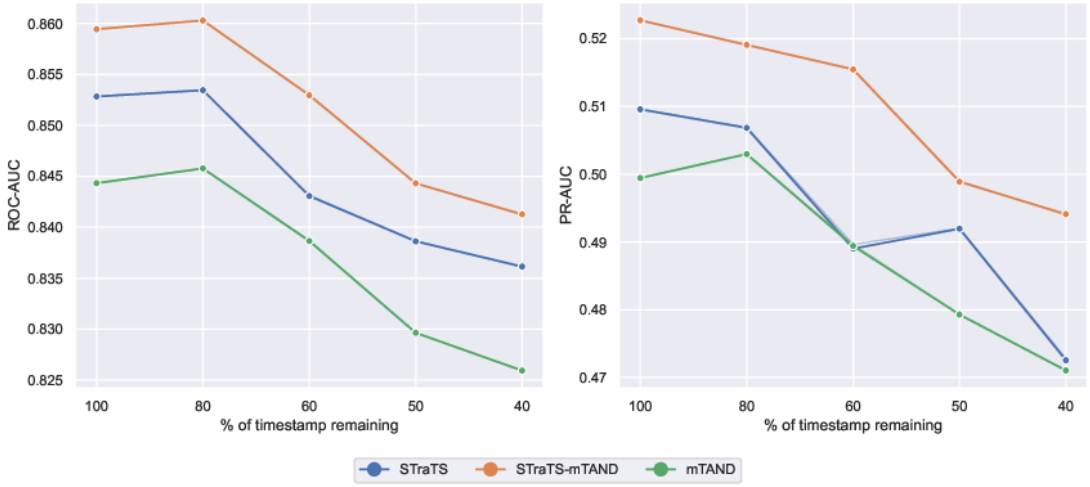

Figure 4.9: Average mortality prediction performance using datasets of different sparsity

In this section, all results reported were conducted using STraTS-mTAND, trained using all labelled data and the results are averaged over 10 runs.

4.6.1 Importance of Different Time Window

To try and identify which time window provides the most explanation on the time-series, we manipulated the dataset to filter for observations within a certain time window. Table

4.10 shows the aggregated statistics of the three datasets after filtering for the respective time window.

Starting Hour	Ending Hour	Avg time between observations(Hours)	Median # of observations	Median # of timestamps
0	12	0.491	123	22
12	24	0.633	101	17
0	24	0.562	228	40
24	36	0.695	96	15
12	36	0.674	199	33
0	36	0.607	329	56
36	48	0.727	91	15
24	48	0.723	189	31
12	48	0.699	294	48
0	48	0.638	424	72

Table 4.10: Statistics of filtered dataset

Table 4.11 shows the results according to the different time intervals. The **best** performance is bold and underlined, **2nd best** is bold and 3rd best performance is underlined.

Starting Hour	Ending Hour	ROC-AUC	PR-AUC
0	12	0.805 ± 0.003	0.399 ± 0.015
12	24	0.800 ± 0.003	0.420 ± 0.007
0	24	0.828 ± 0.005	0.445 ± 0.007
24	36	0.817 ± 0.002	0.467 ± 0.006
12	36	0.829 ± 0.002	0.481 ± 0.004
0	36	**0.849 ± 0.004**	**0.491 ± 0.013**
36	48	0.818 ± 0.004	0.469 ± 0.009
24	48	0.842 ± 0.003	0.514 ± 0.009
12	48	0.843 ± 0.003	0.519 ± 0.006
0	48	**0.860 ± 0.004**	**0.520 ± 0.014**

Table 4.11: STraTS-mTAND mortality prediction performance using different time interval

When the model is trained on 12 consecutive hours of data, using 0-12 hours or 12-24 hours seems to perform worse than using 24-36 hours or 36-48 hours. Figure 4.10 highlights this difference. This is despite the fact that the number of observations in each interval is approximately equal. This suggests to us that observations taken at a later time are more valuable than those taken earlier.

Instead when trained on 24 consecutive hours of data, as shown in Figure 4.11, there is a significant difference in performance when using 24-48 hours as compared to the other

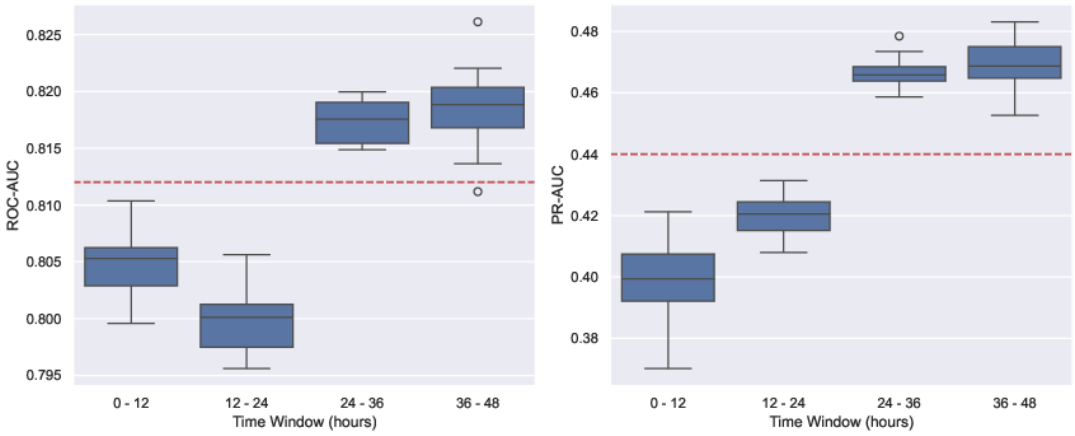

Figure 4.10: Boxplot of mortality prediction performance using 12 hours data

time window. This is in line with what we observed in Figure 4.10, where the 24-36 window and the 36-48 window perform better than the 0-12 and 12-24 window. The increase in performance when combining 24-36 hours and 36-48 hours, also suggests that the information provided by the two time windows is orthogonal and provides additional information for the model.

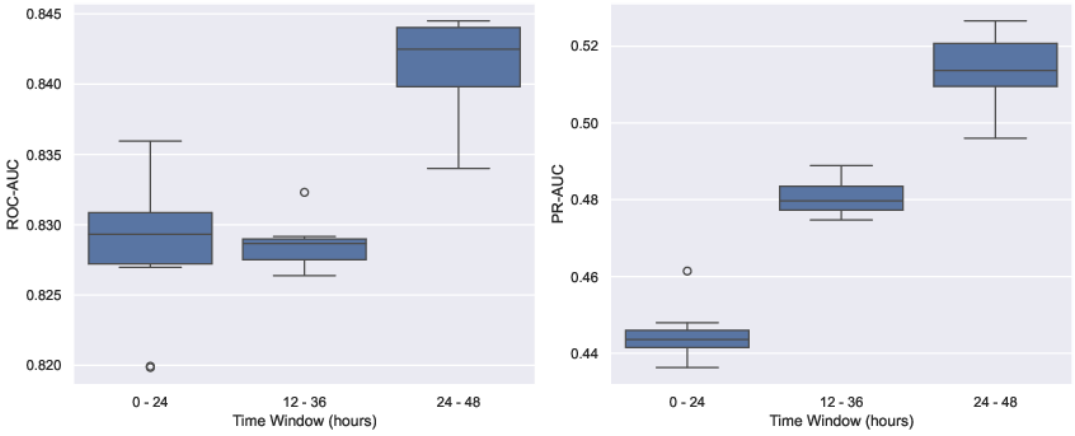

Figure 4.11: Boxplot of mortality prediction performance using 24 hours data

Figure 4.12 shows the results with an increasing time window, starting from 0 hours to 48 hours. As expected, we see an increasing trend in the performance due to more observations per time-series.

Figure 4.13 shows the trend of the performance with an increasing time window, starting

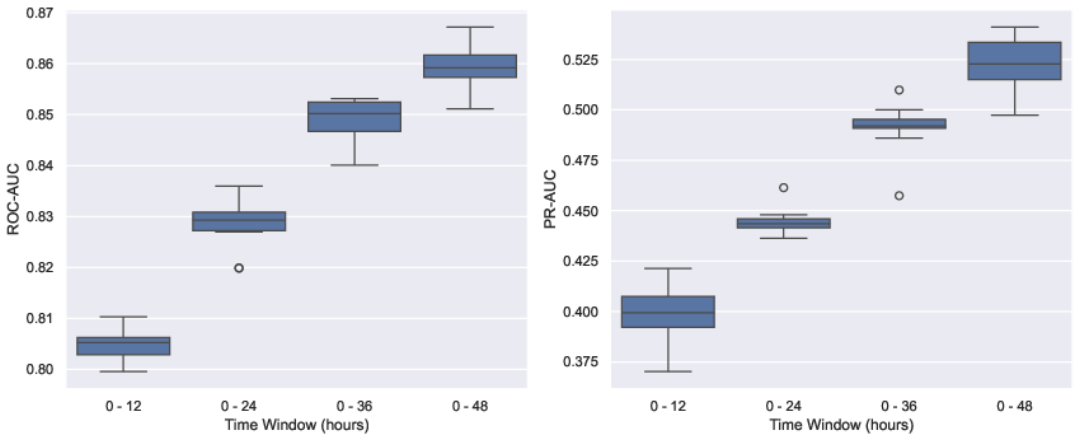

Figure 4.12: Mortality prediction performance with increasing time window, starting from 0 hours to 48 hours

from 48 hours to 0 hours. As expected, we see an increasing trend as well. However, an interesting observation is when comparing 12-48 hours to 24-48 hours, the performance increase is marginal. There is a more significant increase when including the 0-12 hours time window subsequently.

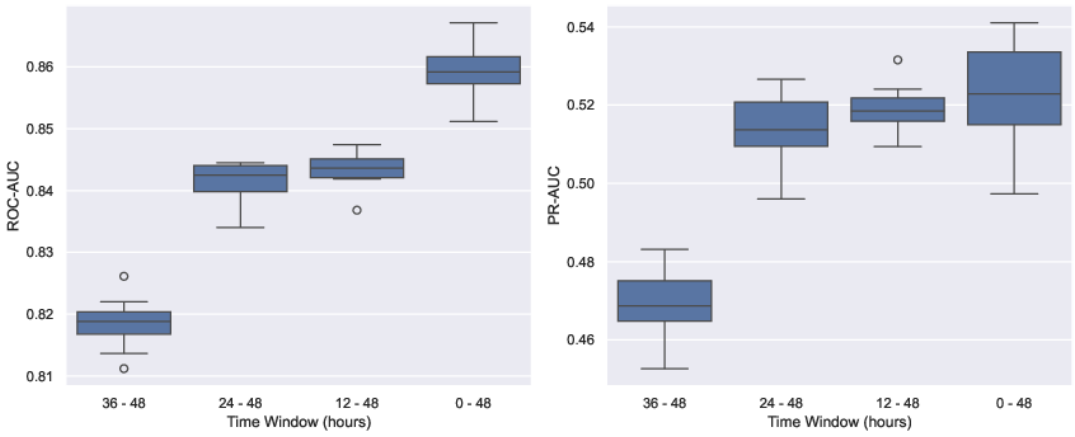

Figure 4.13: Mortality prediction performance with increasing time window, starting from 48 hours to 0 hours

From the findings in Figure 4.13, we followed up by training the model using a time window of 0-12 and 24-48 hours. Table 4.12 shows us that without the data from 12-24 hours, the performance is comparable when using the entire dataset. It seems that using the 0-12 hours time window itself does not provide much information as the

observations are not recent. However, combining it with data from the 24-48 hours time window drastically increases performance. This shows that not all observations are useful in this prediction problem. We hypothesise that closer time intervals between observations do not necessarily provide a good overview of the trajectory of the patients condition. Instead, there is an appropriate time interval that is sufficient for the model to estimate the trajectory and any closer observations may not lead to a better estimation.

Time Window(Hours)	ROC-AUC	PR-AUC
0-12 & 24-48	0.858 ± 0.002	**0.525 ± 0.010**
0-48	**0.860 ± 0.004**	0.520 ± 0.014

Table 4.12: STraTS-mTAND mortality prediction performance

4.6.2 Importance of Static Variables

We removed the static variables from the dataset to try to understand how the static variables are affecting the model prediction abilities. We obtained the results by averaging the results over 10 runs. The results are shown in Table 4.13 with the **best** performance in bold. Without static variables, the ROC-AUC declined by 1.1%. This suggests to us that the static variables do aid in the prediction task.

	ROC-AUC	PR-AUC
With Static Variables	**0.860 ± 0.004**	**0.520 ± 0.014**
Without Static Variables	0.849 ± 0.003	0.504 ± 0.011

Table 4.13: Mortality prediction performance with and without static variables averaged over 10 runs.

To determine how the static variable affects the predicted probability, we plotted the predicted probability density distribution for both models against the different variables. The probability of a sample is obtained by averaging the predicted probability over 10 runs.

Figure 4.14 shows the distribution categorised by age with each group comprising 25% of the test dataset. We observed that for younger patients, aged 14 to 52, the model trained with static variables tends to predict a lower probability of mortality than the model trained without. The opposite goes for older patients, aged 78 to 90, where the probability of mortality tends to be higher from the model trained with static variables.

This is aligned with our intuition that younger patients are more likely to survive given the same medical condition.

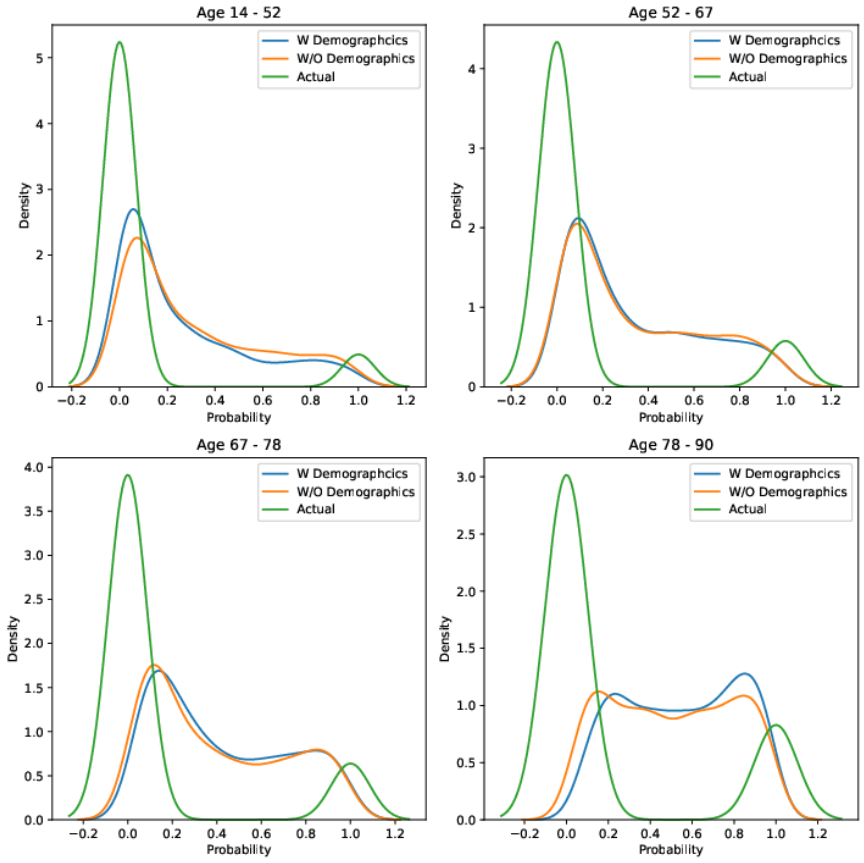

Figure 4.14: Average Predicted Probability from models with and without static variables

Diving into the probability densities when broken down by the ICU types as shown in Figure 4.15, we observe that models trained with static variables tend to fit the actual class distribution of each ICU type slightly better. An example is that for the Cardiac Surgery Recovery Unit, there is a higher proportion of in-hospital survival (class 0) as compared to the other ICU types. This results in the model trained with static variables to have a higher density at lower probabilities of 0.1 and lower densities at 0.7 and above.

In general, without static variables, the model does seem to be able to generate a similar probability distribution as models trained with static variables. However, the static variables are still useful as they shift the probability distribution slightly to fit the actual class distribution better as shown in the better performance in table 4.13

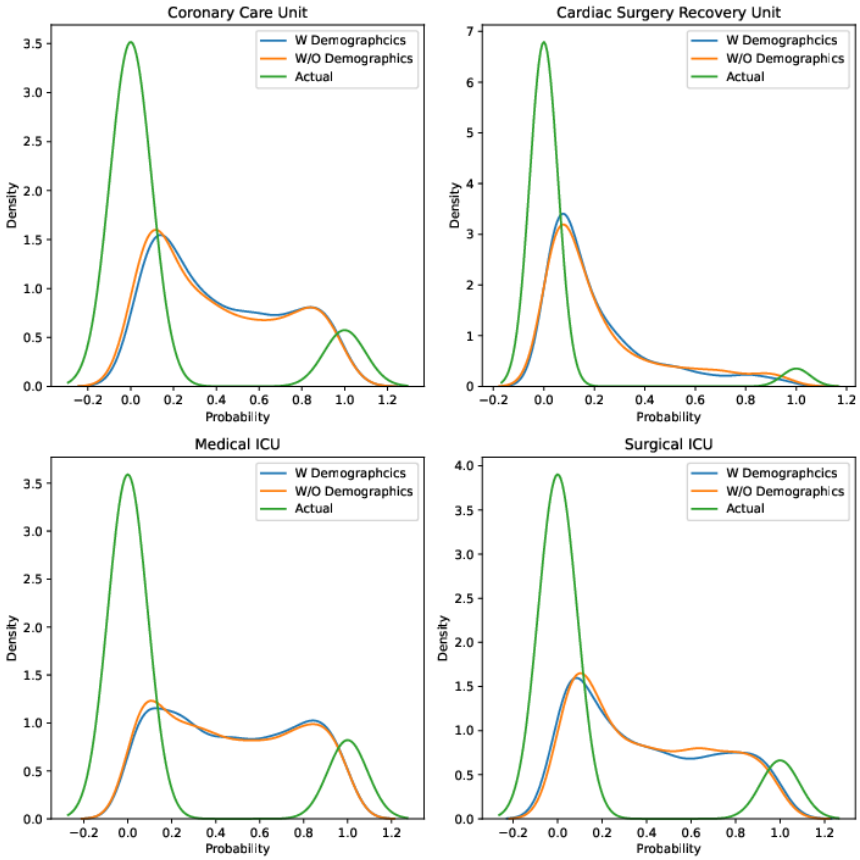

Figure 4.15: Average Predicted Probability from models with and without static variables

Chapter 5

Conclusion and Future Work

Many techniques have been researched to utilise irregularly sampled time-series data to perform predictions in the medical domain. These techniques are mainly grouped into interpolation-based and non-interpolation-based. Both have their merits and flaws. In this study, we introduced a new model, STraTS-mTAND, that combines STraTS, a non-interpolation-based model, and mTAND, an interpolation-based model. Through the combination of both techniques, we can generate a better representation of the time-series and provide more information to the prediction layer. In our paper, we tested our model on the in-hospital mortality prediction problem using the PhysioNet Challenge 2012 and MIMIC-III dataset. Through our experiments, we showed that our model outperforms other interpolation or non-interpolation techniques in the in-hospital mortality problem in terms of the ROC-AUC and PR-AUC metrics. This performance holds when trained with lesser data as well as with sparser time-series datasets, hence showing the robustness of our model.

Further works can focus on experimenting with the time query in the mTAND module. Our experiments used a fixed length and regular intervals for the time query provided to the mTAND module. More work can be done to explore how different variations (i.e. different lengths, different time intervals) of the time query affect the performance as mentioned in section 4.5.3. We hope our work will help medical workers identify high-risk patients ti deliver medical care in a timely fashion.

Bibliography

[1] Julia Adler-Milstein et al. “Electronic Health Record Adoption In US Hospitals: Progress Continues, But Challenges Persist”. en. In: *Health Aff (Millwood)* 34.12 (Nov. 2015), pp. 2174–2180.

[2] D J Cullen et al. “Preventable adverse drug events in hospitalized patients: a comparative study of intensive care and general care units”. en. In: *Crit Care Med* 25.8 (Aug. 1997), pp. 1289–1297.

[3] Cao Xiao, Edward Choi, and Jimeng Sun. “Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review”. en. In: *J Am Med Inform Assoc* 25.10 (Oct. 2018), pp. 1419–1428.

[4] Philip B. Weerakody et al. “A review of irregular time series data handling with gated recurrent neural networks”. In: *Neurocomputing* 441 (2021), pp. 161–178. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.02.046. URL: https://www.sciencedirect.com/science/article/pii/S0925231221003003.

[5] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons, 2019.

[6] Satya Narayan Shukla and Benjamin Marlin. “Multi-Time Attention Networks for Irregularly Sampled Time Series”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=4c0J6lwQ4_.

[7] Sindhu Tipirneni and Chandan K. Reddy. “Self-Supervised Transformer for Sparse and Irregularly Sampled Multivariate Clinical Time-Series”. In: *ACM*

Trans. Knowl. Discov. Data 16.6 (July 2022). ISSN: 1556-4681. DOI: 10.1145/3516367. URL: https://doi.org/10.1145/3516367.

[8] Ikaro Silva et al. "Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012". en. In: *Comput Cardiol (2010)* 39 (2012), pp. 245–248.

[9] Alistair E W Johnson et al. "MIMIC-III, a freely accessible critical care database". In: *Scientific Data* 3.1 (May 2016), p. 160035.

[10] Pranjul Yadav et al. "Mining Electronic Health Records (EHRs): A Survey". In: *ACM Comput. Surv.* 50.6 (Jan. 2018). ISSN: 0360-0300. DOI: 10.1145/3127881. URL: https://doi.org/10.1145/3127881.

[11] Zachary C Lipton, David Kale, and Randall Wetzel. "Directly Modeling Missing Data in Sequences with RNNs: Improved Classification of Clinical Time Series". In: *Proceedings of the 1st Machine Learning for Healthcare Conference*. Ed. by Finale Doshi-Velez et al. Vol. 56. Proceedings of Machine Learning Research. Northeastern University, Boston, MA, USA: PMLR, Aug. 2016, pp. 253–270. URL: https://proceedings.mlr.press/v56/Lipton16.html.

[12] Matthew McDermott et al. "A comprehensive EHR timeseries pre-training benchmark". In: *Proceedings of the Conference on Health, Inference, and Learning*. CHIL '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 257–278. ISBN: 9781450383592. DOI: 10.1145/3450439.3451877. URL: https://doi.org/10.1145/3450439.3451877.

[13] Vincent Fortuin et al. *GP-VAE: Deep Probabilistic Time Series Imputation*. 2020. arXiv: 1907.04155 [stat.ML].

[14] Federico Cismondi et al. "Missing data in medical databases: Impute, delete or classify?" In: *Artificial Intelligence in Medicine* 58.1 (2013), pp. 63–72. ISSN: 0933-3657. DOI: https://doi.org/10.1016/j.artmed.2013.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0933365713000055.

[15] Zhengping Che et al. "Recurrent Neural Networks for Multivariate Time Series with Missing Values". In: *Scientific Reports* 8.1 (Apr. 2018), p. 6085.

[16] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[17] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[18] Trang Pham et al. "Predicting healthcare trajectories from medical records: A deep learning approach". In: *Journal of Biomedical Informatics* 69 (2017), pp. 218–229. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2017.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1532046417300710.

[19] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. *Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences*. 2016. arXiv: 1610.09513 [cs.LG].

[20] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[21] Max Horn et al. "Set Functions for Time Series". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 4353–4363.

[22] Lin Zehui et al. *DropAttention: A Regularization Method for Fully-Connected Self-Attention Networks*. 2019. arXiv: 1907.11065 [cs.CL].

[23] Mohammed Saeed et al. "Multiparameter Intelligent Monitoring in Intensive Care II: a public-access intensive care unit database". en. In: *Crit Care Med* 39.5 (May 2011), pp. 952–960.

[24] Hrayr Harutyunyan et al. "Multitask learning and benchmarking with clinical time series data". In: *Scientific Data* 6.1 (2019), p. 96. ISSN: 2052-4463. DOI: 10.1038/s41597-019-0103-9. URL: https://doi.org/10.1038/s41597-019-0103-9.

[25] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[26] Mohammad Reza Rezaei-Dastjerdehei, Amirmohammad Mijani, and Emad Fatemizadeh. “Addressing Imbalance in Multi-Label Classification Using Weighted Cross Entropy Loss Function”. In: *2020 27th National and 5th International Iranian Conference on Biomedical Engineering (ICBME)*. 2020, pp. 333–338. DOI: 10.1109/ICBME51989.2020.9319440.

[27] Karimollah Hajian-Tilaki. “Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation”. en. In: *Caspian J Intern Med* 4.2 (2013), pp. 627–635.

[28] Takaya Saito and Marc Rehmsmeier. “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets”. en. In: *PLoS One* 10.3 (Mar. 2015), e0118432.

[29] Jesse Davis and Mark Goadrich. “The Relationship Between Precision-Recall and ROC Curves”. In: vol. 06. June 2006. DOI: 10.1145/1143844.1143874.