Self-supervised Transformer for Multivariate Clinical Time-Series with Missing Values

Sindhu Tipirneni
Department of Computer Science
Virginia Tech
tsaisindhura@vt.edu

Chandan K. Reddy
Department of Computer Science
Virginia Tech
reddy@cs.vt.edu

Abstract

Multivariate time-series (MVTS) data are frequently observed in critical care settings and are typically characterized by excessive missingness and irregular time intervals. Existing approaches for learning representations in this domain handle such issues by either aggregation or imputation of values, which in-turn suppresses the fine-grained information and adds undesirable noise/overhead into the machine learning model. To tackle this challenge, we propose STraTS (**S**elf-supervised **Tra**nsformer for **T**ime-**S**eries) model which bypasses these pitfalls by treating time-series as a set of observation triplets instead of using the traditional dense matrix representation. It employs a novel Continuous Value Embedding (CVE) technique to encode continuous time and variable values without the need for discretization. It is composed of a Transformer component with Multi-head attention layers which enables it to learn contextual triplet embeddings while avoiding problems of recurrence and vanishing gradients that occur in recurrent architectures. Many healthcare datasets also suffer from the limited availability of labeled data. Our model utilizes self-supervision by leveraging unlabeled data to learn better representations by performing time-series forecasting as a self-supervision task. Experiments on real-world multivariate clinical time-series benchmark datasets show that STraTS shows better prediction performance than state-of-the-art methods for mortality prediction, especially when labeled data is limited. Finally, we also present an interpretable version of STraTS which can identify important measurements in the time-series data.

1 Introduction

Time-series data routinely occurs in critical care settings where various measurements are recorded for patients throughout their course of stay (Figure 1). Predicting clinical outcomes like mortality, decompensation, length of stay, and disease risk from such complex multi-variate data can facilitate both effective management of critical care units and automatic personalized treatment recommendation for patients. The successes of deep learning in image and text domains realized by CNNs, RNNs (Sutskever et al., 2014; Chung et al., 2014), and Transformers (Vaswani et al., 2017) have inspired the application of these architectures to develop better prediction models for time-series data as well. However, time-series in the clinical domain portray a unique set of challenges that are described below.

- **Missingness and Sparsity**: A patient's condition may demand observing only a subset of variables of interest. Thus, not all the variables are observed for every patient. Also, the observed time-series matrices are very sparse as some variables may be measured more frequently than others for a given patient.
- **Irregular time intervals and Sporadicity**: Clinical variables are not usually measured at regular time intervals. Thus, the measurements occur sporadically in time depending on the underlying condition of the patient.
- **Limited labeled data**: Patient-level clinical data is often expensive to obtain and labeled data subsets pertaining to a specific prediction task may be even more limited (for e.g. building a severity classifier for Covid-19 patients.)

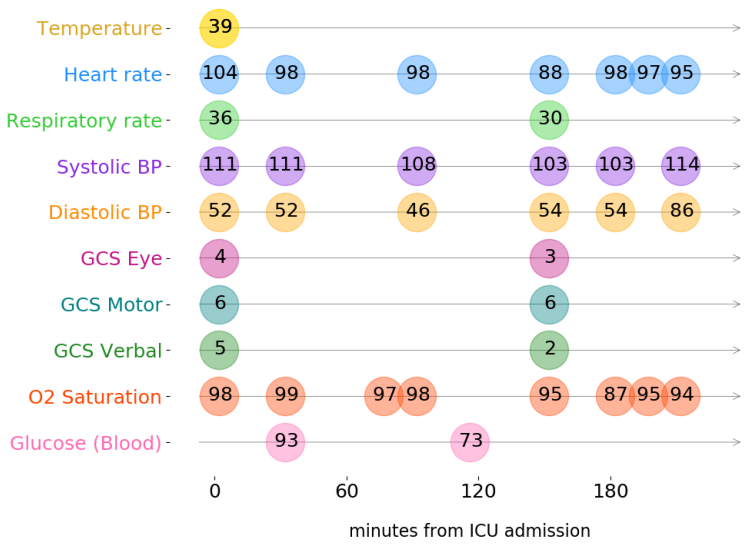

Figure 1: An illustrative example of a multivariate clinical time-series data with irregular time points and missing values.

A straight-forward approach to deal with irregular time intervals and missingness is to aggregate measurements into discrete time intervals and add missingness indicators respectively. However, this suppresses important fine-grained information because the granularity of observed time-series may differ from patient to patient based on the underlying medical condition. Existing sequence models for clinical time-series (Che et al., 2018) and other interpolation-based models (Shukla and Marlin, 2019) address this issue by learnable imputation or interpolation strategies. Such techniques add undesirable noise and extra overhead to the model which usually worsens as the time-series become increasingly sparse. It is also unreasonable to impute clinical variables without careful consideration of the domain knowledge about each variable which is nontrivial to obtain.

Considering these shortcomings, we design a framework that does not need to perform any such operations and directly builds a model *based on only the observations that are available in the data.* Thus, unlike conventional approaches which view each time-series as a matrix of dimensions #features x #time-steps, our model regards each time-series as a set of observation triplets (a triple containing time, variable, and value). The proposed STraTS (which stands for **S**elf-supervised **Tra**nsformer for **T**ime-**S**eries) model embeds these triplets by using a novel Continuous Value Embedding (CVE) scheme to avoid the need for binning continuous values before embedding them. The use of CVE for embedding time preserves the fine grained information which is lost when the time-axis is discretized. STraTS encodes contextual information of observation triplets by using a Transformer-based architecture with multi-head attention. We choose this over recurrent neural network (RNN) architectures because the sequential nature of RNN models hinders parallel processing while the Transformer bypasses this by using self-attention to attend from every token to every other token in a single step.

To build superior representations using limited labeled data, we employ self-supervision and develop a time-series forecasting task to pretrain STraTS. This enables learning generalized representations in the presence of limited labeled data and alleviates sensitivity to noise. Furthermore, interpretable models are usually preferred in healthcare but existing deep models for clinical time-series lack this attribute. Thus, we also propose an interpretable version of our model (I-STraTS) which slightly compromises on performance metrics but can identify important measurements in the input. Though we evaluate the proposed model only on binary classification tasks, note that the framework can be utilized in other supervised and unsupervised settings as well, where learning robust and generalized representations of sparse sporadic time-series is desired. The main contributions of our work are summarized below.

- We propose a Transformer-based architecture called STraTS for clinical time-series which addresses the unique characteristics of missingness and sporadicity of such data by avoiding aggregation and imputation.
- We propose a novel Continuous Value Embedding (CVE) mechanism using a one-to-many feed-forward network to embed continuous times and measured values in order to preserve fine grained information.
- STraTS utilizes forecasting as a self-supervision task to leverage unlabeled data to learn more generalized and robust representations.
- We also propose an interpretable version of STraTS that can be used when this is more desired compared to quantitative performance gains.
- Experiments demonstrate that the design choices of STraTS lead to its better performance over competitive baseline models for mortality prediction on two real-world datasets.

The rest of the paper is organized as follows. In section 2, we review relevant literature about tackling sparse and sporadic time-series data, and self-supervised learning. Section 3 formally defines the prediction problem and gives a detailed description of the architecture of STraTS along with the self-supervision approach. Section 4 presents experimental results comparing STraTS with various baselines and demonstrates the interpretability of I-STraTS with a case study. Finally, section 5 concludes the paper and provides future directions.

2 Related Work

2.1 Clinical Time-Series

A straightforward approach to address missing values and irregular time intervals is to impute and aggregate the time-series respectively, before feeding them to a classifier. However, such classifiers ignore the missingness in data which can be quite informative. Lipton et al. (2016) show that phenotyping performance can be improved by passing missingness indicators as additional features to an RNN classifier.

Several early works rely on Gaussian Processes (GP) (Rasmussen, 2003) to model irregular time-series. For example, Lu et al. (2008) represent each time-series as a smooth curve in a RKHS using GP by optimizing GP parameters using Expectation Maximization(EM), and then derive a distance measure on the RKHS which is used to define the SVM classifier's kernel. To account for uncertainty in GP which is ignored in the former, Li and Marlin (2015) formulate the kernel by applying an uncertainty-aware base kernel (called the expected Gaussian kernel) to a series of sliding windows. These works take a two-step approach by first optimizing GP parameters and then training the classification model. To enable end-to-end training, Li and Marlin (2016) again represent time-series using GP posterior at predefined time points but use the reparametrization trick by back-propagating the gradients through a black-box classifier (learnable by gradient-descent) into the GP model. The end-to-end model is uncertainty-aware as the output is formulated as a random variable as well. Futoma et al. (2017) extend this idea to multivariate time-series with the help of multitask GP (Bonilla et al., 2008) to consider inter-variable similarities. Though GP provide a systematic way to deal with uncertainty, they are expensive to learn and their flexibility is limited by the choice of covariance and mean functions.

Shukla and Marlin (2019) also propose an end-to-end method that constitutes interpolation and classification networks stacked in a sequence. However, the learnable interpolation layers approximate the time-series at regular predefined time points in a deterministic fashion (unlike GP-based methods) and allow information sharing across both time and variable dimensions.

Other approaches modify traditional recurrent architectures for clinical time-series to deal with missing values and/or irregular time intervals. For example, Baytas et al. (2017) developed a time-aware long-short term memory (T-LSTM) which is a modification of the LSTM cell to adjust hidden state according to the irregular time gaps. ODE-RNN (Rubanova et al., 2019) uses ODEs to model the continuous-time dynamics of the hidden state while also updating the hidden state at each observed time point using a standard GRU cell. The GRU-D (Che et al., 2018) model is a modification of the GRU cell which decays inputs (to global means) and hidden states through unobserved time intervals. DATA-GRU (Tan et al., 2020), in addition to

decaying the GRU hidden state according to elapsed time, also employs a dual attention mechanism based on missingness and imputation reliability to process inputs before feeding them to a GRU cell.

The imputation/interpolation schemes in the models discussed above can lead to excessive computations and unnecessary noise particularly when missing rates are quite high. Our model is designed to circumvent this issue by representing sparse and irregular time-series as a set of observations. Horn et al. (2019) develop SeFT with a similar idea and use a parametrized set function for classification. The attention-based aggregation used in SeFT contains the same queries for all observations to facilitate low memory and time complexity while compromising on accuracy. The initial embedding in SeFT contains fixed time encodings while our approach uses learnable embeddings for all the three components (time, variable, value) of the observation triplet.

The challenge of training in scenarios with limited labeled data still remains. In order to address this issue, we turn towards self-supervision in order to better utilize the available data to learn effective representations.

2.2 Self-supervised learning

It is well known that the more data that is available to the deep learning model, the more generalized and robust its learned representations are. Limited data can make the model easily overfit to training data and make the model more sensitive to noise. As labeled data is expensive to obtain, self-supervised learning was introduced as a technique to solve this challenge by constructing proxy tasks using a semi-automatic label generation process (Liu et al., 2020). Though this technique has shown great performance boosts with image (Jing and Tian, 2020) and text (Devlin et al., 2019; Yang et al., 2019) data, its application to time-series data has been limited. One such effort is made by Jawed et al. (2020) who use a 1D CNN for dense univariate time-series classification and show increased accuracy by using forecasting as an additional task in a muti-task learning framework. In our work, we also demonstrate time-series forecasting as a viable and effective self-supervision task. Our work is the first to explore self-supervised learning in the context of sparse and irregular multivariate time-series.

3 Proposed Approach

In this section, we describe our STraTS model by first introducing the problem with relevant notation and definitions and then explaining the different components of the model which are illustrated in Figure 3.

3.1 Problem Definition

As stated in the previous sections, STraTS represents a time-series as a set of observation triplets. Formally, an *observation triplet* is defined as a triple (t, f, v) where $t \in \mathbb{R}_{\geq 0}$ is the time, $f \in \mathcal{F}$ is the feature/variable, and $v \in \mathbb{R}$ is the value of the observation. A *multivariate time-series* $\mathbf{T}$ of length n is a defined as a set of n observation triplets i.e., $\mathbf{T} = \{(t_i, f_i, v_i)\}_{i=1}^{n}$.

Consider a dataset $\mathcal{D} = \{(\mathbf{d}^k, \mathbf{T}^k, y^k)\}_{k=1}^{N}$ with N labeled samples, where the k^{th} sample contains a demographic vector $\mathbf{d^k} \in \mathbb{R}^D$, a multivariate time-series $\mathbf{T^k}$, and a corresponding binary label $y^k \in \{0, 1\}$. In this work, each sample corresponds to a single ICU stay where several clinical variables of the patient are measured at irregular time intervals and the binary label indicates in-hospital mortality. The underlying set of time-series variables denoted by $\mathcal{F}$ may include vitals (such as temperature), lab measurements (such as hemoglobin), and input/output events (such as fluid intake and urine output). Thus, the *target task* aims to predict y^k given $(\mathbf{d}^k, \mathbf{T}^k)$.

Our model also incorporates forecasting as a self-supervision task. For this task, we consider a bigger dataset with $N' \geq N$ samples given by $\mathcal{D}' = \{(\mathbf{d}^k, \mathbf{T}^k,$
$\mathbf{m}^k, \mathbf{z}^k)\}_{k=1}^{N'}$. Here, $\mathbf{m^k} \in \mathbb{R}^{|\mathcal{F}|}$ is the forecast mask which indicates whether each variable was observed in the forecast window and $\mathbf{z^k} \in \mathbb{R}^{|\mathcal{F}|}$ contains the corresponding variable values. The forecast mask is necessary because the unobserved forecasts cannot be used in training and are hence masked out in the loss function. The time-series in this dataset are obtained from both the labeled and unlabeled time-series by considering different observation windows. Figure 2 illustrates the construction of inputs and outputs for the target task and forecast task.

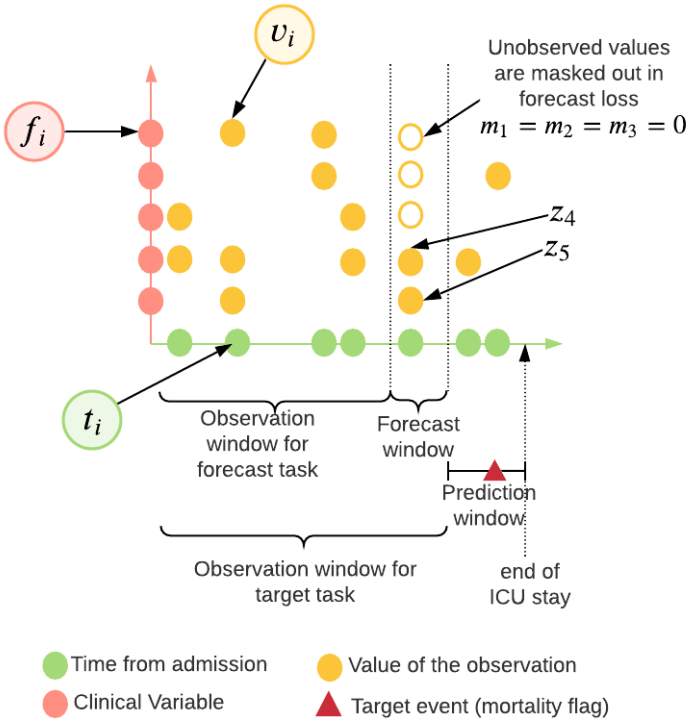

Figure 2: An illustration of input and output construction for target and self-supervision (forecast) tasks. Note that there are several possibilities for segmenting the time axis for forecast task and only one such possibility is shown here.

3.2 The Proposed STraTS Architecture

The architecture of STraTS is illustrated in Figure 3. Unlike most existing approaches which take time-series matrix as the input, STraTS *defines its input as a set of observation triplets*. Each observation triplet in the input is embedded using the Initial Triplet Embedding module. The initial triplet embeddings are then passed through a Contextual Triplet Embedding module which utilizes the Transfomer architecture to encode the context for each triplet. The Fusion Self-attention module then combines these contextual embeddings via self-attention mechanism to generate an embedding for the input time-series which is concatenated with demographics embedding and passed through a feed-forward network to make the final prediction. The notations used in the paper are summarized in Table 1.

3.2.1 Initial Triplet Embedding

Given an input time-series $\mathbf{T} = \{(t_i, f_i, v_i)\}_{i=1}^{n}$, the initial embedding for the i^{th} triplet $\mathbf{e_i} \in \mathbb{R}^d$ is computed by summing the following component embeddings: (i) *Feature embedding* $\mathbf{e_i^f} \in \mathbb{R}^d$, (ii) *Value embedding* $\mathbf{e_i^v} \in \mathbb{R}^d$, and (iii) *Time embedding* $\mathbf{e_i^t} \in \mathbb{R}^d$. In other words, $\mathbf{e_i} = \mathbf{e_i^f} + \mathbf{e_i^v} + \mathbf{e_i^t} \in \mathbb{R}^d$. Feature embeddings $\mathbf{e^f}(.)$ are obtained from a simple lookup table just like word embeddings. Since feature values and times are continuous unlike feature names which are categorical objects, we cannot use a lookup table to embed these continuous values unless they are categorized. Some researchers (Vaswani et al., 2017; Yin et al., 2020) have used sinusoidal encodings to embed continuous values. We propose a novel continuous value embedding (CVE) technique using a one-to-many Feed-forward Network(FFN) with learnable parameters i.e. $\mathbf{e_i^v} = FFN^v(v_i)$, and $\mathbf{e_i^t} = FFN^t(t_i)$.

Both the FFNs have one input neuron and d output neurons and a single hidden layer with $\lfloor\sqrt{d}\rfloor$ neurons and $tanh(.)$ activation. They are of the form $FFN(x) = U\,tanh(Wx + b)$ where the dimensions of weights

Table 1: Notations used in this paper.

Notation	Definition
N	# Time-series for target task
N'	# Time-series for forecast task
$\mathbf{d} \in \mathbb{R}^D$	Demographics vector
$\mathcal{F}$	Set of clinical variables
$t_i \in \mathbb{R}_{\geq 0}$	Time of i^{th} observation
$f_i \in \mathcal{F}$	Variable of i^{th} observation
$v_i \in \mathbb{R}$	Value of i^{th} observation
(t, f, v)	Observation triplet
$\mathbf{T} = \{(t_i, f_i, v_i)\}_{i=1}^n$	Multivariate time-series
$y, \tilde{y} \in \mathbb{R}^{d_o}$	True and predicted outputs for target task
$\mathbf{z}, \tilde{\mathbf{z}} \in \mathbb{R}^{\lvert\mathcal{F}\rvert}$	True and predicted outputs for forecast task
$\mathbf{m} \in \mathbb{R}^{\lvert\mathcal{F}\rvert}$	Forecast mask
$\mathbf{e}_i^t$, $\mathbf{e}_i^v \in \mathbb{R}^d$	CVE for time and value
$\mathbf{e}_i^f \in \mathbb{R}^d$	Variable embedding
$\mathbf{e}_i \in \mathbb{R}^d$	Initial triplet embedding
$\mathbf{e}^T \in \mathbb{R}^d$	Time-series embedding
$\mathbf{e}^d \in \mathbb{R}^d$	Demographics embedding

$\{W, b, U\}$ can be inferred from the size of hidden and output layers. Unlike sinusoidal encodings with fixed frequencies, this technique offers more flexibility by allowing end-to-end learning of continuous value and time embeddings without the need to categorize them.

3.2.2 Contextual Triplet Embedding

The initial triplet embeddings $\{\mathbf{e}_1, ..., \mathbf{e}_n\}$ are then passed through a Transformer architecture (Vaswani et al., 2017) with M blocks, each containing a Multi-Head Attention (MHA) layer with h attention heads and an FFN with one hidden layer. Each block takes n input embeddings $\mathbf{E} \in \mathbb{R}^{n \times d}$ and outputs the corresponding n output embeddings $\mathbf{C} \in \mathbb{R}^{n \times d}$ that capture contextual information. MHA layers use multiple attention heads to attend to information contained in different embedding projections in parallel. The computations of the MHA layer are given by

$$\mathbf{H}_j = softmax\left(\frac{(\mathbf{E}\mathbf{W}_j^q)(\mathbf{E}\mathbf{W}_j^k)^T}{\sqrt{d/h}}\right)(\mathbf{E}\mathbf{W}_j^v) \quad j = 1, ..., h$$
$$MHA(\mathbf{E}) = concat(\mathbf{H}_1, ..., \mathbf{H}_h)\,\mathbf{W}_c$$

Each head projects the input embeddings into query, key, and value subspaces using matrices $\{\mathbf{W}_j^q, \mathbf{W}_j^k, \mathbf{W}_j^v\} \subset \mathbb{R}^{d \times d/h}$. The queries and keys are then used to compute attention weights which are used to compute weighted averages of values. Finally, the outputs of all heads are concatenated and projected to original dimension with $\mathbf{W}_c \in \mathbb{R}^{hd \times d}$. The FFN layer takes the form

$$\mathbf{F}(\mathbf{X}) = ReLU(\mathbf{X}\mathbf{W}_1^f + \mathbf{b}_1^f)\,\mathbf{W}_2^f + \mathbf{b}_2^f$$

with weights $\mathbf{W_1^f} \in \mathbb{R}^{d \times 2d}$, $\mathbf{b_1^f} \in \mathbb{R}^{2d}$, $\mathbf{W_2^f} \in \mathbb{R}^{2d \times d}$, $\mathbf{b_2^f} \in \mathbb{R}^d$. Dropout, residual connections, and layer normalization are added for every MHA and FFN layer. Also, attention dropout randomly masks out some positions in the attention matrix before the softmax computation during training. The output of each block is fed as input to the succeeding one, and the output of the last block gives the contextual triplet embeddings $\{\mathbf{c}_1, ..., \mathbf{c}_n\}$.

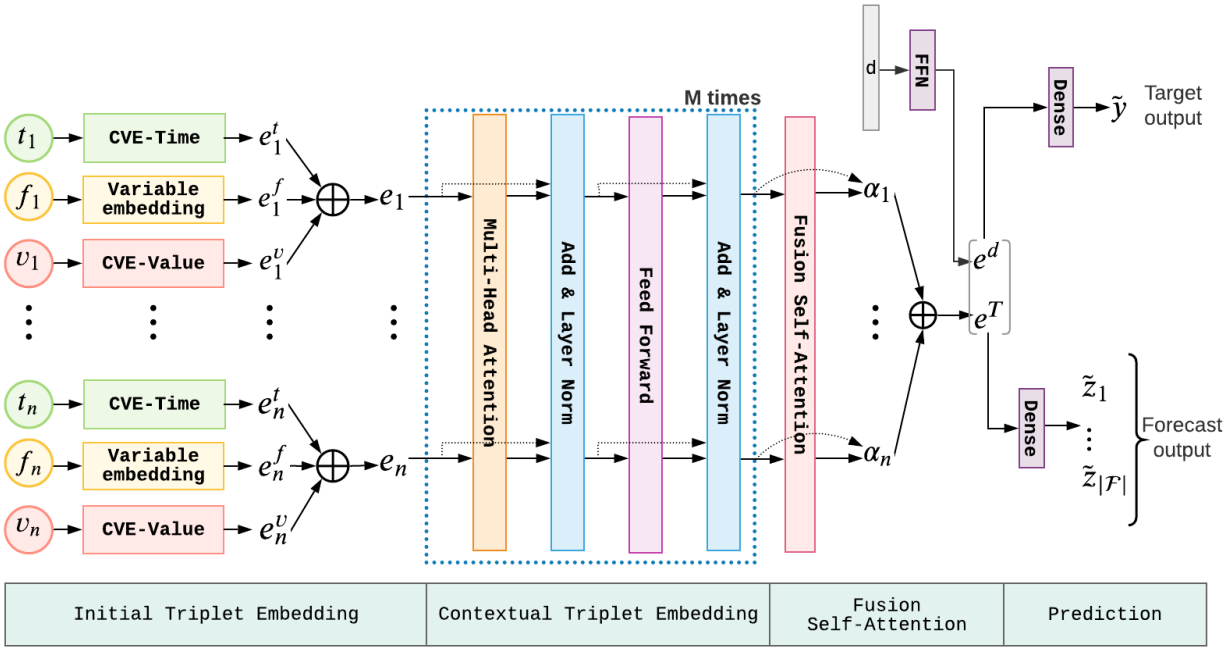

Figure 3: The overall architecture of the proposed STraTS model. The Input Triplet Embedding module embeds each observation triplet, the Contextual Triplet Embedding module encodes contextual information for the triplets, the Fusion Self-Attention module computes times series embedding which is concatenated with demographics embedding and passed through a dense layer to generate predictions for target and self-supervision (forecast) tasks.

3.2.3 Fusion Self-attention

After computing contextual embeddings using a Transformer, we fuse them using a self-attention layer to compute time-series embedding $\mathbf{e}^T \in \mathbb{R}^d$. This layer first computes attention weights $\{\alpha_1, ..., \alpha_n\}$ by passing each contextual embedding through an FFN and computing a softmax over all the FFN outputs.

$$a_i = \mathbf{u}_a^T \, tanh(\mathbf{W}_a \mathbf{c}_i + \mathbf{b}_a)$$
$$\alpha_i = \frac{exp(a_i)}{\sum_{j=1}^{n} exp(a_j)} \quad \forall i = 1, ..., n$$

$\mathbf{W}_a \in \mathbb{R}^{d_a \times d}, \mathbf{b}_a \in \mathbb{R}^{d_a}, \mathbf{u_a} \in \mathbb{R}^{d_a}$ are the weights of this attention network which has d_a neurons in the hidden layer. The time-series embedding is then computed as

$$\mathbf{e}^T = \sum_{i=1}^{n} \alpha_i \mathbf{c_i}$$

3.2.4 Demographics Embedding

We realize that demographics can be encoded as triplets with a default value for time. However, we found that the prediction models performed better in our experiments when demographics are processed separately as follows by passing $\mathbf{d}$ through an FFN. The demographics embedding is thus obtained as

$$\mathbf{e}^d = tanh(\mathbf{W}_2^d \, tanh(\mathbf{W}_1^d \mathbf{d} + \mathbf{b}_1^d) + \mathbf{b}_2^d) \in \mathbb{R}^d$$

where the hidden layer has dimension $2d$.

3.2.5 Prediction Head

The final prediction for target task is obtained by passing the concatenation of demographics and time-series embeddings through a dense layer with sigmoid activation.

$$\tilde{y} = sigmoid(\mathbf{w}_o^T[\mathbf{e}^d \quad \mathbf{e}^T] + b_o)$$

The model is trained on the target task using cross-entropy loss.

3.2.6 Self-supervision

We experimented with both masking and forecasting as pretext tasks for providing self-supervision and found that forecasting improved the results. The forecasting task uses the same architecture as the target task except for the prediction layer i.e.

$$\tilde{\mathbf{z}} = \mathbf{W}_s[\mathbf{e}^d \quad \mathbf{e}^T] + \mathbf{b}_s \in \mathbb{R}^{|\mathcal{F}|}$$

A masked MSE loss is used for training on the forecast task to account for missing values in the forecast outputs. Thus, the loss for self-supervision is given by

$$\mathcal{L}_{ss} = \frac{1}{|N'|} \sum_{k=1}^{N'} \sum_{j=1}^{|\mathcal{F}|} m_j^k (\tilde{\mathbf{z}}_j^k - \mathbf{z}_j^k)^2$$

where $\mathbf{m}_j^k = 1$ (or $\mathbf{m}_j^k = 0$) if the ground truth forecast $\mathbf{z}_j^k$ is available (or unavailable) for j^{th} variable in k^{th} sample. The model is first pretrained on the self-supervision task and is then fine-tuned on the target task.

3.3 Interpretability

We also propose of an interpretable version of our model which we refer to as I-STraTS. Inspired by Choi et al. (2016) and Zhang et al. (2020), we alter the architecture of STraTS in such a way that the output can be expressed using a linear combination of components that are derived from individual features. Specifically, the output of I-STraTS is formulated as

$$\tilde{y} = sigmoid\left(\mathbf{w}_o^T[\mathbf{d} \quad \sum_{i=1}^{n} \alpha_i \mathbf{e}_i] + b_o\right)$$

Contrary to STraTS, (i) we combine the initial triplet embeddings using the attention weights in Fusion Self-attention module, and (ii) directly use the raw demographics vector as the demographics embedding. The above equation can also be written as

$$\tilde{y} = sigmoid\left(\sum_{j=1}^{D} \mathbf{w}_o[j]\, \mathbf{d}[j] + \sum_{i=1}^{n} \sum_{j=1}^{d} \alpha_i\, \mathbf{w}_o[j+D]\, \mathbf{e}_i[j] + b_o\right) \tag{1}$$

Thus, we assign a 'contribution score' to the j^{th} demographic feature as $\mathbf{w}_o[j]\mathbf{d}j$ and to the j^{th} time-series observation as $\sum_{j=1}^{d} \alpha_i\, \mathbf{w}_o[j+D]\, \mathbf{e}_i[j]$.

4 Experiments

We evaluated our proposed STraTS model against state-of-the-art baselines on two real-world EHR databases for the mortality prediction task. This section starts with a description of the datasets and baselines, followed by a discussion of results focusing on generalization and interpretability.

Table 2: Basic dataset statistics. (Avg. variable missing rate and Avg. # observations/stay are calculated using supervised samples only.)

	MIMIC-III	PhysioNet-2012
# ICU stays	52,871	11,988
# ICU stays (supervised)	44,812	11,988
# Avg. span of time-series	101.9h	47.3h
# Avg. span of time-sries (supervised)	23.5h	47.3h
# Variables	129	37
Avg. variable missing rate	89.7%	79.7%
Avg. # observations/stay	401	436
Demographics	Age, Gender	Age, Gender, Height, ICU Type
Task	24-hour mortality	48-hour mortality
% positive class	9.7%	14.2%

4.1 Datasets

We experiment with time-series extracted from two real-world EHR datasets which are described below. The dataset statistics are summarised in Table 2.

MIMIC-III (Johnson et al., 2016): This is a publicly available database containing medical records of about $46k$ critical care patients in Beth Israel Deaconess Medical Center between 2001 and 2012. We filtered ICU stays to include only adult patients and extracted 129 features from the following tables: input events, output events, lab events, chart events, and prescriptions for each ICU stay. For mortality prediction task, we only include ICU stays that lasted for atleast one day with the patient alive at the end of first day, and predict in-hospital mortality using the first 24 hours of data. For forecasting, the set of observation windows is defined (in hours) as $\{[min(0, x-24), x) \mid 20 \leq x \leq 124,\ x\%4 = 0\}$ and the prediction window is the 2-hour period following the observation window. Note that we only consider those samples which have atleast one time-series measurement in both observation and prediction windows. The data is split at patient level into training, validation, and test sets in the ratio $64:16:20$.

PhysioNet Challenge 2012: This processed dataset from Physionet Challenge 2012 [1] contains records of $11,988$ ICU stays of adult patients. The target task aims to predict in-hospital mortality given the first 48 hours of data for each ICU stay. Since demographic variables 'gender' and 'height' are not available for all ICU stays, we perform mean imputation and add missingness indicators for them as additional demographic variables. To generate inputs and outputs for forecasting, the set of observation windows is defined (in hours) as $\{[0, x) \mid 12 \leq x \leq 44,\ x\%4 = 0\}$ and the prediction window is the 2-hour period following the observation window. The data from set-b and set-c together is split into training and validation (80:20) while set-a is used for testing.

4.2 Baseline Methods

To demonstrate the effectiveness of STraTS over the state-of-the-art, we compare it with the following baseline methods.

- **GRU** (Chung et al., 2014): The input is a time-series matrix with hourly aggregation where missing variables are mean-imputed. Binary missingness indicators and time (scaled to [0,1]) since the last observation of each variable are also included as additional features at each time step. The final hidden state is transformed by a dense layer to generate output.
- **TCN** (Bai et al., 2018): This model takes the same input as GRU which is passed through a stack of temporal convolution layers with residual connections. The representation from the last time step of the last layer is transformed by a dense layer to generate output.

[1] https://physionet.org/content/challenge-2012/1.0.0/

Table 3: Mortality prediction performance on MIMIC-III and PhysioNet-2012 datasets. The results show mean and standard deviation of metrics after repeating the experiment 10 times by sampling 50% labeled data each time.

		ROC-AUC	PR-AUC	min(Re,Pr)
MIMIC-III	GRU	0.886 ± 0.002	0.559 ± 0.005	0.533 ± 0.007
	TCN	0.879 ± 0.001	0.540 ± 0.004	0.525 ± 0.005
	SAnD	0.876 ± 0.002	0.533 ± 0.011	0.515 ± 0.008
	GRU-D	0.883 ± 0.003	0.544 ± 0.007	0.527 ± 0.005
	InterpNet	0.881 ± 0.002	0.540 ± 0.007	0.516 ± 0.005
	SeFT	0.881 ± 0.003	0.547 ± 0.011	0.524 ± 0.01
	STraTS	$\mathbf{0.891 \pm 0.002}$	$\mathbf{0.577 \pm 0.006}$	$\mathbf{0.541 \pm 0.008}$
PhysioNet-2012	GRU	0.831 ± 0.003	0.468 ± 0.008	0.465 ± 0.009
	TCN	0.813 ± 0.005	0.430 ± 0.01	0.433 ± 0.009
	SAnD	0.800 ± 0.013	0.406 ± 0.021	0.418 ± 0.018
	GRU-D	0.833 ± 0.005	0.481 ± 0.008	0.468 ± 0.012
	InterpNet	0.822 ± 0.007	0.460 ± 0.017	0.455 ± 0.017
	SeFT	0.832 ± 0.005	0.454 ± 0.017	0.465 ± 0.009
	STraTS	$\mathbf{0.839 \pm 0.008}$	$\mathbf{0.498 \pm 0.012}$	$\mathbf{0.483 \pm 0.01}$

- **SaND** (Song et al., 2018): This model also has the same input representation as GRU and the input is passed through a Transformer with causal attention and a dense interpolation layer.
- **GRU-D** (Che et al., 2018): The GRU-D cell takes a vector of variable values at each time one/more measurements are seen. The GRU-D cell, which is a modification to the GRU cell, decays unobserved values in this vector to global mean values and also adjusts the hidden state according to elapsed times since the last observation of each variable.
- **InterpNet** (Shukla and Marlin, 2019): This model consists of a semi-parametric interpolation network that interpolates all variables at regular predefined time points, followed by a prediction network which is a GRU. It also uses a reconstruction loss to enhance the interpolation network. The input representation is similar to that of GRU-D and therefore, no aggregation is performed.
- **SeFT** (Horn et al., 2019): This model also inputs a set of observation triplets, similar to STraTS. It uses sinusoidal encodings to embed times and the deep network used to combine the observation embeddings is formulated as a set function using a simpler but faster variation of multi-head attention.

For all the baselines, we use two dense layers to get the demographics encoding and concatenate it to the time-series representation before the last dense layer. All the baselines use sigmoid activation at the last dense layer for mortality prediction. The time-series measurements (by variable) and demographics vectors are normalized to have zero mean and unit variance. All models are trained using the Adam (Kingma and Ba, 2015) optimizer. More implementation details are provided in the appendix.

4.3 Evaluation Metrics

The following metrics are used to quantitatively compare the baselines and proposed models for the binary classification task of mortality prediction.

- ROC-AUC: Area under ROC curve.
- PR-AUC: Area under precision-recall curve.
- min(Re, Pr): This metric is computed as the maximum of 'minimum of recall and precision' across all thresholds.

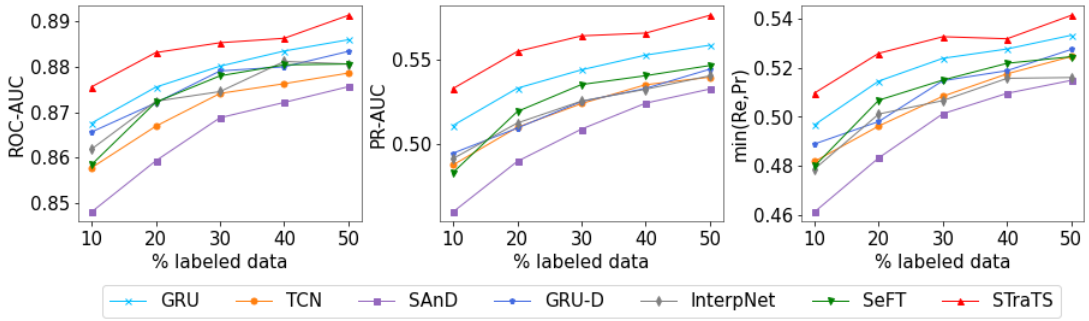

Figure 4: Mortality prediction performance on MIMIC-III for different percentages of labeled data averaged over 10 runs.

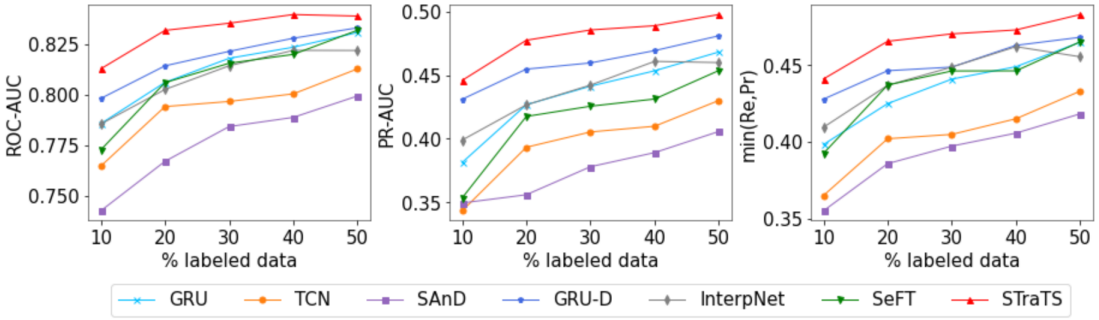

Figure 5: Mortality prediction performance on PhysioNet-2012 dataset for different percentages of labeled data averaged over 10 runs.

4.4 Prediction Performance

We train each model using 10 different random samplings of 50% labeled data from the train and validation sets. Note that STraTS uses the entire labeled data and additional unlabeled data (if available) for self-supervision. Table 3 shows the results for mortality prediction on MIMIC-III and PhysioNet-2012 datasets which are averaged over the 10 runs. STraTS achieves the best performance on all metrics, improving PR-AUC by 1.8% and 1.7% on MIMIC-III and PhysioNet-2012 datasets respectively over the best baseline. This shows that our design choices of triplet embedding, attention-based architecture, and self-supervision enable STraTS to learn superior representations. We expected the interpolation-based models GRU-D and InterpNet to outperform the simpler models GRU, TCN, and SaND. This was true for all cases except that GRU showed a better performance than GRU-D and InterpNet on the MIMIC-III dataset, which needs to be investigated further.

To test the generalization ability of different models, we evaluate STraTS and the baseline models by training them on varying percentages of labeled data. Lower proportions of labeled data can be observed in the real-world when there are several right-censored samples. Figures 4 and 5 show the results for MIMIC-III and PhysioNet-2012 datasets, respectively. The performance of all models declines with reduced labeled data. But STraTS is seen to have a crucial advantage compared to other models in lower labeled data settings which can be attributed to self-supervision.

4.5 Ablation Study

We compared the predictive performance of STraTs and I-STraTS, with and without self-supervision. The results are shown in Table 4. 'ss+' and 'ss-' are used to indicate models trained with and without self-supervision respectively. We observe that: (i) Adding interpretability to STraTS hurts the prediction scores as a result of constraining model representations. (ii) Adding self-supervision improves performance of both

Table 4: Ablation Study: Comparing mortality prediction performance of STraTS and I-STraTS with and without self-supervision. ('ss+' and 'ss-' are used to indicate models trained with and without self-supervision respectively.)

		ROC-AUC	PR-AUC	min(Re,Pr)
MIMIC-III	I-STraTS (ss-)	0.878 ± 0.002	0.542 ± 0.006	0.516 ± 0.008
	I-STraTS (ss+)	0.887 ± 0.003	0.556 ± 0.008	0.531 ± 0.005
	STraTS (ss-)	0.881 ± 0.002	0.546 ± 0.007	0.525 ± 0.012
	STraTS (ss+)	$\mathbf{0.891 \pm 0.002}$	$\mathbf{0.577 \pm 0.006}$	$\mathbf{0.541 \pm 0.008}$
PhysioNet-2012	I-STraTS (ss-)	0.826 ± 0.008	0.456 ± 0.018	0.467 ± 0.025
	I-STraTS (ss+)	0.833 ± 0.007	0.478 ± 0.015	0.466 ± 0.010
	STraTS (ss-)	0.835 ± 0.009	0.467 ± 0.023	0.471 ± 0.017
	STraTS (ss+)	$\mathbf{0.839 \pm 0.008}$	$\mathbf{0.498 \pm 0.012}$	$\mathbf{0.483 \pm 0.010}$

Table 5: Case study: Top variables ordered by 'contribution score' for an ICU stay from MIMIC-III dataset.

Variable	Range/Value	'contribution score'
Hematocrit	[28.7, 30.8]	0.448
Age	85.0	0.395
Phosphate	[2.7, 3.5]	0.237
RBC	[3.0, 3.1]	0.116
MCV	[95.0, 98.0]	0.087
MCHC	[32.5, 33.9]	0.077
Potassium	[3.8, 4.7]	0.072
Bilirubin (Total)	[0.7, 0.8]	0.065

STraTS and I-STraTS. (iii) I-STraTS(ss+) outperforms STraTS(ss-) on all metrics on MIMIC-III dataset, and on the PR-AUC metric for PhysioNet-2012 dataset. This demonstrates that the performance drop from introducing interpretability can be compensated by the performance gains of self-supervision.

4.6 Interpretability

To illustrate how I-STraTS explains its predictions, we present a case study for an 85 year old female patient from the MIMIC-III dataset who expired on the 6^{th} day after ICU admission. The model I-STraTS predicts the probability of her in-hospital mortality as 0.95 using data collected just on the first day. The patient had 382 measurements belonging to 58 time-series variables. The top 8 variables ordered by their 'cumulative 'contribution score' along with the range (if multiple observations) or value (if one observation) are shown in Table 5. We see that I-STraTS considers the abnormal Hematocrit values and old age as the most important observations in predicting that the patient is at high risk of mortality. Such predictions can not only guide the healthcare system in identifying high-risk patients for better resource allocation but also guide the clinicians into understanding the contributing factors and make better diagnoses and treatment choices.

To get a more fine-grained intuition, the observed time-series for some variables in this ICU stay are plotted in Figure 6 along with the corresponding contribution scores. It is interesting to see that the contribution scores appear to be positively or negatively correlated with the underlying values. For example, as Hct decreases, the model gives more weight to the measurement. Similarly, as GCS-eye increases, the model pays less attention to it. Higher FiO2 implies that the patient is under ventilation and is hence considered important. The contribution scores of BP time-series also exhibit a pattern. Lower and more recent values of SBP and DBP contribute more towards the final prediction.

5 Conclusion

We proposed a Transformer-based model, STraTS, for prediction tasks on multivariate clinical time-series to address the challenges faced by existing methods in this domain. Our approach of using observation triplets

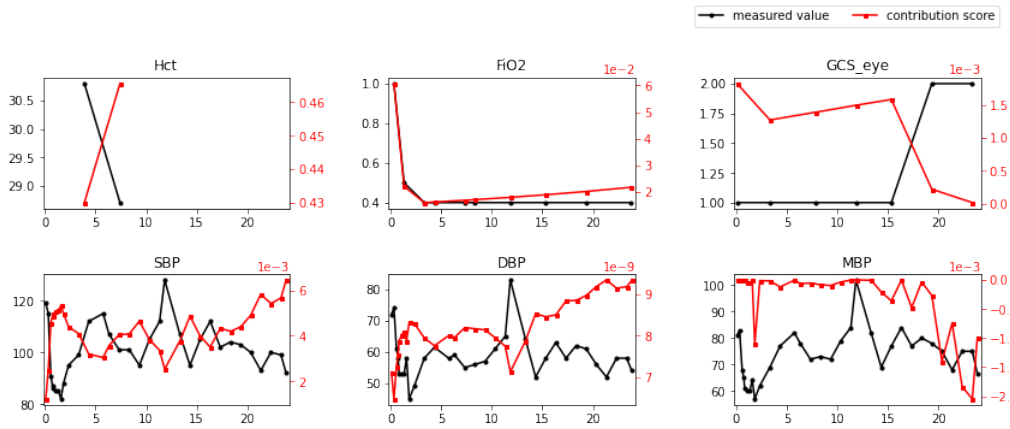

Figure 6: Case study: An illustration of few time-series with contribution scores for a patient from MIMIC-III dataset.

as time-series components avoids the problems faced by aggregation and imputation methods for sparse and sporadic multivariate time-series. We leave it for future work to develop heuristics to quantify the gains of triplet-based representations over aggregation and interpolation based ones, in terms of accuracy and time-and-space complexity, based on the degree of sparsity and sporadicity in data. The self-supervision task of forecasting using unlabeled data enables STraTS to learn more generalized representations, thus outperforming state-of-the-art baselines. This motivates us to explore the effectiveness of more self-supervision tasks for clinical time-series data. We also proposed an interpretable version of STraTS called I-STraTS for which self-supervision compensates the drop in prediction performance from introducing interpretability.

References

Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR abs/1803.01271, 1803.01271

Baytas IM, Xiao C, Zhang X, Wang F, Jain AK, Zhou J (2017) Patient subtyping via time-aware lstm networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 65–74

Bonilla EV, Chai KM, Williams C (2008) Multi-task gaussian process prediction. In: Advances in neural information processing systems, pp 153–160

Che Z, Purushotham S, Cho K, Sontag D, Liu Y (2018) Recurrent neural networks for multivariate time series with missing values. Scientific reports 8(1):1–12

Choi E, Bahadori MT, Sun J, Kulas J, Schuetz A, Stewart W (2016) Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In: Advances in Neural Information Processing Systems, vol 29

Chung J, Gülçehre Ç, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR abs/1412.3555, 1412.3555

Devlin J, Chang M, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the NAACL-HLT, Volume 1, pp 4171–4186, DOI 10.18653/v1/n19-1423

Futoma J, Hariharan S, Heller KA (2017) Learning to detect sepsis with a multitask gaussian process RNN classifier. In: Proceedings of the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 70, pp 1174–1182

Horn M, Moor M, Bock C, Rieck B, Borgwardt KM (2019) Set functions for time series. CoRR abs/1909.12064, 1909.12064

Jawed S, Grabocka J, Schmidt-Thieme L (2020) Self-supervised learning for semi-supervised time series classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 499–511

Jing L, Tian Y (2020) Self-supervised visual feature learning with deep neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence

Johnson AE, Pollard TJ, Shen L, Li-wei HL, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, Mark RG (2016) Mimic-iii, a freely accessible critical care database. Scientific data 3:160035

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, Conference Track Proceedings

Li SCX, Marlin BM (2015) Classification of sparse and irregularly sampled time series with mixtures of expected gaussian kernels and random features. In: UAI, pp 484–493

Li SCX, Marlin BM (2016) A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In: Advances in neural information processing systems, pp 1804–1812

Lipton ZC, Kale D, Wetzel R (2016) Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In: Machine Learning for Healthcare Conference, pp 253–270

Liu X, Zhang F, Hou Z, Wang Z, Mian L, Zhang J, Tang J (2020) Self-supervised learning: Generative or contrastive. CoRR abs/2006.08218, 2006.08218

Lu Z, Leen TK, Huang Y, Erdogmus D (2008) A reproducing kernel hilbert space framework for pairwise time series distances. In: Proceedings of the 25th international conference on Machine learning, pp 624–631

Rasmussen CE (2003) Gaussian processes in machine learning. In: Summer school on machine learning, Springer, pp 63–71

Rubanova Y, Chen RT, Duvenaud DK (2019) Latent ordinary differential equations for irregularly-sampled time series. In: Advances in Neural Information Processing Systems, pp 5320–5330

Shukla SN, Marlin BM (2019) Interpolation-prediction networks for irregularly sampled time series. CoRR abs/1909.07782, 1909.07782

Song H, Rajan D, Thiagarajan JJ, Spanias A (2018) Attend and diagnose: Clinical time series analysis using attention models. In: Thirty-second AAAI conference on artificial intelligence

Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp 3104–3112

Tan Q, Ye M, Yang B, Liu S, Ma AJ, Yip TCF, Wong GLH, Yuen P (2020) Data-gru: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 34, pp 930–937

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008

Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov RR, Le QV (2019) Xlnet: Generalized autoregressive pretraining for language understanding. In: Advances in neural information processing systems, pp 5753–5763

Yin C, Liu R, Zhang D, Zhang P (2020) Identifying sepsis subphenotypes via time-aware multi-modal auto-encoder. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 862–872

Zhang X, Qian B, Cao S, Li Y, Chen H, Zheng Y, Davidson I (2020) Inprem: An interpretable and trustworthy predictive model for healthcare. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 450–460

A APPENDIX

A.1 Implementation details

Table 6 lists the hyperparameters used in the experiments for all models for MIMIC-III and PhysioNet-2012 datasets. All models are trained using a batch size of 32 with Adam optimizer and training is stopped when sum of ROC-AUC and PR-AUC does not improve for 10 epochs. For pretraining phase using the self-supervision task, the patience is set to 5 epochs and epoch size is set to $256,000$ samples. For MIMIC-III dataset, we set the maximum number of time-steps for GRU-D and InterpNet, and the maximum no. of observations for STraTS using the 99^{th} percentile for the same. This is done to avoid memory overflow during batch training. The deep models are implemented using keras with tensorflow backend. For InterpNet, we adapted the official code from `https://github.com/mlds-lab/interp-net`. For GRU-D and SeFT, we borrowed implementations from `https://github.com/BorgwardtLab/Set_Functions_for_Time_Series`. The experiments are conducted on a single NVIDIA GRID P40-12Q GPU. The implementation is publicly available at `https://github.com/sindhura97/STraTS`.

Model	MIMIC-III	PhysioNet-2012
GRU	units=50, rec d/o=0.2, output d/o=0.2, lr=0.0001	units=43 rec d/o=0.2, output d/o=0.2, lr=0.0001
TCN	layers=4, filters=128, kernel size=4, d/o=0.1, lr=0.0001	layers=6, filters=64, kernel size=4, d/o=0.1, lr=0.0005
SAnD	N=4, r=24, M=12, d/o=0.3, d=64, h=2, he=8, lr=0.0005	N=4, r=24, M=12, d/o=0.3, d=64, h=2, he=8, lr=0.0005
GRU-D	units=60, rec d/o=0.2, output d/o=0.2, lr=0.0001	units=49 rec d/o=0.2, output d/o=0.2, lr=0.0001
SeFT	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	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
InterpNet	ref points=96, units=100, input d/o=0.2, rec d/o=0.2, lr=0.001	ref points=192, units=100, input d/o=0.2, rec d/o=0.2, lr=0.001
STraTS(ss-) & I-STraTS(ss-)	d=32, M=2, h=4, d/o=0.2, lr=0.0005	d=32, M=2, h=4, d/o=0.2, lr=0.001
STraTS & I-STraTS	d=50, M=2, h=4, d/o=0.2, lr=0.0005	d=50, M=2, h=4, d/o=0.2, lr=0.0005

Table 6: Hyperparameters used for experiments in this paper.