IVP-VAE: Modeling EHR Time Series with Initial Value Problem Solvers

Jingge Xiao[1], Leonie Basso[1], Wolfgang Nejdl[1], Niloy Ganguly[2], Sandipan Sikdar[1]

[1]L3S Research Center, Leibniz Universität Hannover
[2]Indian Institute of Technology Kharagpur
{xiao, basso, nejdl, sandipan.sikdar}@L3S.de, niloy@cse.iitkgp.ac.in

Abstract

Continuous-time models such as Neural ODEs and Neural Flows have shown promising results in analyzing irregularly sampled time series frequently encountered in electronic health records. Based on these models, time series are typically processed with a hybrid of an initial value problem (IVP) solver and a recurrent neural network within the variational autoencoder architecture. Sequentially solving IVPs makes such models computationally less efficient. In this paper, we propose to model time series purely with continuous processes whose state evolution can be approximated directly by IVPs. This eliminates the need for recurrent computation and enables multiple states to evolve in parallel. We further fuse the encoder and decoder with one IVP solver utilizing its invertibility, which leads to fewer parameters and faster convergence. Experiments on three real-world datasets show that the proposed method can systematically outperform its predecessors, achieve state-of-the-art results, and have significant advantages in terms of data efficiency.

Introduction

Electronic Health Record (EHR) data contains multi-variate time series of patient information, such as vital signs and laboratory results, which can be utilized to perform diagnosis or recommend treatment (McDermott et al. 2021). The data in EHR time series is often irregularly sampled (i.e., unequal time intervals between successive measurements) and can have missing values (Zhang et al. 2022). The irregularity is caused mainly due to unstructured manual processes, event-driven recordings, device failure, and also different sampling frequencies across multiple variables (Weerakody et al. 2021). These complexities make learning and modeling clinical time series data particularly challenging for classical machine learning models (Shukla and Marlin 2021b; Sun et al. 2020). In recent years, significant progress has been made in the development of models for handling irregularly sampled time series data (Che et al. 2018; Rubanova, Chen, and Duvenaud 2019; Shukla and Marlin 2021a; Zhang et al. 2022), which have been extensively tested on EHR data.

Neural ODEs (Chen et al. 2018) are continuous-time models based on ordinary differential equations (ODEs) that can naturally handle irregularly sampled data. The data is assumed to be generated by a continuous process that is modeled using ODEs.

Rubanova, Chen, and Duvenaud (2019) further extend the idea and develop Latent-ODE by integrating Neural ODEs and recurrent neural network (RNN) into a variational autoencoder (VAE) (Kingma and Welling 2014) architecture. However, neural ODE models require deploying a numerical ODE solver which is computationally expensive. Biloš et al. (2021) hence propose an efficient alternative by directly modeling the solution of ODEs with a neural network, thereby obtaining a variant of Latent-ODE using Neural Flows, referred to as Latent-Flow in this paper. However, when analyzing time series, these Latent-based continuous-time models (Latent-ODE and Latent-Flow) require sequential processing of data, which makes them inefficient and hard to train.

In this work, we propose IVP-VAE, a continuous-time model specifically designed for EHR time series, which is capable of dealing with irregularly sampled time series data in a non-sequential way. Different from Latent-ODE and Latent-Flow, our model takes variational approximation purely as solving initial value problems (IVPs). Specifically, observations at different time points are mapped to states of an unknown continuous process and propagated to a latent variable z_0 by solving different IVPs in parallel. This parallelization leads to a significant speedup over existing continuous-time models. Latent-based continuous-time models use the VAE architecture, whose encoder and decoder consist of separate recognition and generative modules. We observe that neural IVP solvers are inherently invertible, i.e., IVPs can be solved in both forward and backward time directions, and exploit this property to utilize the same solver for both encoding and decoding. Our design results in reduced model complexity in terms of number of parameters and convergence rate.

We deploy our model on the tasks of time series forecasting and classification across three real-world EHR datasets. IVP-VAE generally outperforms the existing latent-based continuous-time models across all the datasets and tasks. More importantly, it achieves more than one order of magnitude speedup over its latent-based predecessors. With regard to the state-of-the-art irregular sampled time series classification and forecasting models, IVP-VAE consistently ranks among the top-2 models, even though the baselines are in many cases task-specific. IVP-VAE offers the best performance-efficiency trade-off across all the tasks. Additionally, our model is able to achieve significant improvements in settings where the training data is limited, which is often encountered in healthcare applications (e.g., cohort of patients with a particular condition). We summarize the main contributions of the current work below -

- We propose a novel continuous-time model IVP-VAE, which can capture sequential patterns of EHR time series by purely solving multiple IVPs in parallel.
- By utilizing the invertibility property of IVP solvers, we achieve parameter sharing between encoder and decoder of the VAE architecture, and thus provide a more efficient generative modeling technique.
- Across real-world datasets on both forecasting and classification tasks, IVP-VAE achieves a higher efficiency compared to the existing continuous-time models. With regard to other state-of-the-art models, it achieves a better performance-efficiency trade-off.
- IVP-VAE achieves significant improvements over baseline models in settings where the training data is limited.

Source code is at https://github.com/jingge326/ivpvae.

Background and Related Work

EHR data contains comprehensive information about patients' health conditions and has empowered the research on developing personalized medicine (Abul-Husn and Kenny 2019). The availability of several large EHR datasets, including MIMIC-III (Johnson et al. 2016), MIMIC-IV (Johnson et al. 2023), and eICU (Pollard et al. 2019), has facilitated the development of deep learning models for this domain. Specific tasks like time series forecasting and mortality prediction have been widely used to test models' capability in data modeling and representation learning (Harutyunyan et al. 2019; McDermott et al. 2021; Purushotham et al. 2018; Schirmer et al. 2022). Functions built upon these can be used to support early warning of deterioration, identify patients at risk, diagnosis, etc. (Gao et al. 2020; Syed et al. 2021). However, EHR time series are usually irregularly sampled (Zhang et al. 2022), i.e., the time interval between consecutive observations is not fixed, and only some or no observations are available at each timestamp, making them sparse and of variable length (Weerakody et al. 2021).

There has been significant progress in developing models that are naturally able to handle irregularly sampled time series as the input (Shukla and Marlin 2021b). Several studies propose recurrent models that add decay mechanisms to model the irregularity in observations while training (Cao et al. 2018; Kim and Chi 2018; Li and Xu 2019). For example, GRU-D (Che et al. 2018) uses a temporal decay mechanism that is based on gated recurrent units (GRUs) and incorporates missing patterns. However, along with recurrent units comes the unstable gradient issue, and difficulties in long sequence modeling and parallelizing (Lipton, Berkowitz, and Elkan 2015). Another group of work introduces attention mechanisms into models for irregular time series (Chien and Chen 2021; Horn et al. 2020; Shukla and Marlin 2021a; Tipirneni and Reddy 2022). For example, Raindrop (Zhang et al. 2022) combines attention with graph neural networks to model irregularity. Owing to quadratic computation complexity and high memory usage, deploying these models to longer sequences becomes practically infeasible (Zhou et al. 2021). Convolutional models for irregular time series formulate the convolutional kernels as continuous functions (Fey et al. 2018; Li and Marlin 2020; Romero et al. 2022), enabling them to handle sequences with arbitrary size and irregular sample intervals. However, when dealing with arbitrary length sequences, they usually need to first pad missing entries with specific values (such as zero) (Romero et al. 2022), which can introduce irrelevant data and conceal important information.

Neural ODEs (Chen et al. 2018) are continuous-time models that can naturally handle irregularly sampled data. The Latent ODE model (Rubanova, Chen, and Duvenaud 2019) uses an ODE-RNN encoder in a VAE (Kingma and Welling 2014) architecture. GRU-ODE-Bayes (De Brouwer et al. 2019) combines ODE and GRU into a continuous-time version of the GRU. Solving an ODE with a numerical ODE-Solver is computationally expensive. Neural Flow (Biloš et al. 2021) proposes an efficient alternative. The solution of an ODE is modelled directly with a neural network instead of using a numerical solver (see methodology section about continuous-time models for details). A shortcoming of current research in this area is that existing methods often require sequentially solving a large amount of ODEs, which makes the training and inference less efficient.

Present Work Our method builds on VAE-based continuous models with Neural ODE and Neural Flow as IVP solvers. We introduce a set of novel architectural designs to further improve the efficiency. An embedding layer maps the input into a latent space where the IVP solvers are deployed. We eliminate the need for recurrent and sequential computation by modeling each time point as an IVP. As the IVP solvers are invertible by design, we propose to use the same IVP solver in the encoder and decoder of a VAE.

Methodology

In this section, we first formulate the problem, followed by a brief background on continuous-time models. We then introduce and describe our model in detail.

Problem Formulation

In our setup, we consider a multivariate time series X as a sequence of L observations: $X = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^{L}$. Each observation $\boldsymbol{x}_i$ is collected at a time t_i. $\boldsymbol{x}_i \in \mathbf{R}^D$ where D represents the number of variables being measured at each time point (e.g., in EHR data these could represent a patient's heart rate, respiratory rate etc.). The dataset $\mathcal{X}$ consists of N such sequences, $\mathcal{X} = \{X_1, \ldots, X_N\}$, collected within a fixed time window. Note that the length L of the sequences can vary across the dataset due to the irregular spacing of the observation time points.

Our goal is to first build a generative model g for irregularly sampled time series (like EHR), which is capable of forecasting future values, and additionally augment it with a classifier to conduct classification tasks for which g serves as a representation learning module. The time series forecasting task is to predict observations X^τ collected in time window $[T, T+\tau]$, based on past observations X, where τ is the forecast horizon. The classification task is to predict the categorical label y of the sample X.

Continuous-time Models

A continuous-time model (Chen et al. 2018) assumes that the data $\boldsymbol{x}_t$ at time t is generated by a latent process F whose state $\boldsymbol{z}$ can be propagated continuously to serve diverse purposes, such as generative modeling or representation learning. The propagation is achieved by solving IVPs, which are ODEs together with initial conditions, i.e.

$$F(t_0) = \boldsymbol{z}_0 \tag{1}$$

$$f(t, \boldsymbol{z}_t) = \frac{dF(t)}{dt} \tag{2}$$

$$\boldsymbol{z}_i = \boldsymbol{z}_{i-1} + \int_{t_{i-1}}^{t_i} f(t, \boldsymbol{z}_t) dt \tag{3}$$

Neural ODEs (Chen et al. 2018) parameterize f with uniformly Lipschitz continuous neural networks, which are used to specify the derivative at every $t \in [t_0, T]$. States of the continuous process were calculated using the Runge–Kutta method or other numeric integrators. Neural Flow (Biloš et al. 2021) proposes to directly model the solution curve F with invertible neural networks. Both Neural ODEs and Neural Flow propagate hidden states by solving IVPs.

These continuous-time models are combined with recurrent units to analyze irregular time series. Given a hidden state $\boldsymbol{z}_{i-1}$, the idea to obtain the next hidden state is to propagate $\boldsymbol{z}$ using an IVP solver until the next observation $\boldsymbol{x}_i$ at time t_i and then to use an RNN cell to update it, as expressed by the following equations.

$$\boldsymbol{z}_i^- = \mathrm{IVPSolve}((\boldsymbol{z}_{i-1}, t_{i-1}), t_i) \tag{4}$$

$$\boldsymbol{z}_i = \mathrm{RNN}(x_t, \boldsymbol{z}_i^-) \tag{5}$$

This represents the general IVP-RNN hybrid model first proposed by Rubanova, Chen, and Duvenaud (2019) using Neural ODEs as the IVP solver, known as Latent-ODE. The Latent-Flow variant of Latent-ODE uses Neural Flow as the IVP solver to directly obtain $\boldsymbol{z}_i^-$ (Biloš et al. 2021). The entire model is trained as a VAE with the IVP-RNN hybrid model used as encoder to infer the posterior. Both Latent-ODE and Latent-Flow have proven to be effective in modeling irregular time series. However, the sequential nature of processing information makes these models computationally less efficient.

Proposed Model: IVP-VAE

The key idea that our model builds upon is that time series values $\{\boldsymbol{x}_i\}_{i=1}^L$ are discrete observations of an unknown continuous process. Each sample X correspondingly represents a continuous process, of which we obtain an indirect observation at each available timestamp t_i. In this sense, our proposed model IVP-VAE is essentially a generative model for these continuous processes. From this idea, we design the model following two basic points: (i) We can circumvent the sequential operation bottleneck by processing all time steps independently as one ODE's different IVPs which can be solved in parallel. (ii) IVP solvers are inherently invertible, which enables us to use the same solver for both forward and backward propagation. The model is trained as a VAE whose encoder includes an embedding module and the IVP solver evolving latent state $\boldsymbol{z}_i$ backward in time, while the decoder includes the same IVP solver evolving the state forward in time, and a reconstruction module generating estimated data $\hat{\boldsymbol{x}}_i$ based on state $\boldsymbol{z}_i$. The model is illustrated in Figure 1 and the whole idea is summarized in Algorithm 1. The steps are described in the following sections. The IVP-VAE model can then be used for different downstream tasks, for example by appending a classification module.

Algorithm 1: IVP-VAE. The same IVP solver works for both encoder and decoder by solving IVPs in opposite directions.

Input: Data points and timestamps $X = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^L$
Output: Reconstructed $\{\hat{\boldsymbol{x}}_i\}_{i=1}^L$

1: $t_0 = 0$
2: $\{\boldsymbol{z}_i\}_{i=1}^L = \mathrm{Embedding}(\{\boldsymbol{x}_i\}_{i=1}^L)$
3: $\{\Delta t_i\}_{i=1}^L = t_0 - \{t_i\}_{i=1}^L$
4: $\{\boldsymbol{z}_0^i\}_{i=1}^L = \{\mathrm{IVPSolve}(\boldsymbol{z}_i, \Delta t_i)\}_{i=1}^L$
5: $q(\boldsymbol{z}_0|X) = \mathrm{Inference}\left(\{\boldsymbol{z}_0^i\}_{i=1}^L\right)$
6: $\boldsymbol{z}_0 \sim q(\boldsymbol{z}_0|X)$

$$\left.\begin{array}{ll} 7: & \{\Delta t_i\}_{i=1}^L = \{t_i\}_{i=1}^L - t_0 \\ 8: & \{\boldsymbol{z}_i\}_{i=1}^L = \{\mathrm{IVPSolve}(\boldsymbol{z}_0, \Delta t_i)\}_{i=1}^L \\ 9: & \{\hat{\boldsymbol{x}}_i\}_{i=1}^L = \mathrm{Reconstruct}(\{\boldsymbol{z}_i\}_{i=1}^L) \end{array}\right\} \; p_\theta(X \mid z_0).$$

10: **return** $\{\hat{\boldsymbol{x}}_i\}_{i=1}^L$

Embedding and Reconstruction Within the embedding module, given a time series $X = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^L$, we first generate corresponding binary masks $\{\boldsymbol{m}_i\}_{i=1}^L$ that indicate which variables are observed and which are not at time t_i. Next, we obtain $\boldsymbol{v}_i = (\boldsymbol{x}_i | \boldsymbol{m}_i)$ for all observations at t_i by concatenating $\boldsymbol{x}_i$ with $\boldsymbol{m}_i$. A neural network ϵ is then deployed on $\boldsymbol{v}_i$, $\boldsymbol{z}_i = \epsilon(\boldsymbol{v}_i)$, to extract useful information from multivariate observations at each timestamp, and produce $\boldsymbol{z}_i$ which represents the state of the continuous process at t_i.

On the decoder side, we design a similar module for data reconstruction that maps $\boldsymbol{z}_i$ to $\boldsymbol{x}_i$. The aim of adding embedding and reconstruction modules is to create a space in which the latent state $\boldsymbol{z}$ evolves, and also re-organize information into a more compact form. For these two modules, we use MLPs for demonstration and brevity. They can be more complex or well-designed networks. The embedding and the reconstruction operation are represented by line 2 and line 9 in Algorithm 1, respectively.

Evolving Backward in Time Given that the true posterior $p(\boldsymbol{z}_0 \mid X)$ is intractable (Kingma and Welling 2014), the overall goal is to approximate the posterior, i.e., learn a variational approximation $q_\phi(\boldsymbol{z}_0 \mid X)$ which can then be used to sample z_0. For $\boldsymbol{x}_i$, the initial condition is defined as $(\boldsymbol{z}_i, t_i)$ in the encoder. The task of a neural IVP solver is to start from t_i, move towards t_0 continuously and calculate $\boldsymbol{z}_0$:

$$\boldsymbol{z}_0^i = \mathrm{IVPSolve}(\boldsymbol{z}_i, \Delta t_i), \tag{6}$$

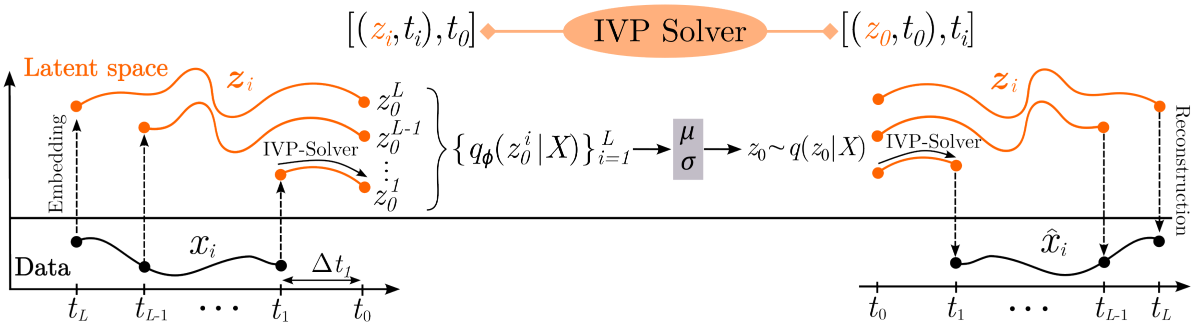

Figure 1: Modeling irregular time series with IVP-VAE. (Left) In the encoder, an embedding module maps data $\boldsymbol{x}_i$ into latent state $\boldsymbol{z}_i$. The state is evolved backward in time: Take $(\boldsymbol{z}_i, t_i)$ as initial condition and calculate state $\boldsymbol{z}_0$ at t_0 using an IVP solver. (Right) In the decoder, the latent state is evolved forward in time: Take $(\boldsymbol{z}_0, t_0)$ as initial condition and go opposite along the timeline to obtain state $\boldsymbol{z}_i$ using the same IVP solver. A reconstruction module then maps $\boldsymbol{z}_i$ back to data $\hat{\boldsymbol{x}}_i$.

where $\Delta t_i = t_0 - t_i$. $(\boldsymbol{z}_i, t_i)$ and $(\boldsymbol{z}_0^i, t_0)$ are on the same integral curve and satisfy the same ODE. Similarly, $\{\boldsymbol{z}_0^i\}_{i=1}^L$ is obtained for all $\{\boldsymbol{x}_i, \Delta t_i\}_{i=1}^L$. As we take observation $\boldsymbol{x}_i$ as an indirect observation of the unknown continuous process, we can make a guess of this process based on each $\boldsymbol{x}_i$, and then derive $\boldsymbol{z}_0^i$ of the process. Here, $\boldsymbol{z}_0^i$ is an estimation of $\boldsymbol{z}_0$ made by the IVP solver based on $\boldsymbol{x}_i$. Afterward, there are two issues to be addressed. First, each $\boldsymbol{z}_0^i$ should approximate $\boldsymbol{z}_0$ during training. Second, all L $\boldsymbol{z}_0^i$ should be integrated together for the following generative module (decoder). For the first issue, we will discuss more details below in the training section. For the second issue, we define $q_\phi(\boldsymbol{z}_0 \mid X)$ to be the posterior distribution over the latent variable $\boldsymbol{z}_0$ induced by the input time series X. To obtain it from $\left\{q_\phi(\boldsymbol{z}_0^i \mid X)\right\}_{i=1}^L$ in Inference (line 5 in Algorithm 1), we introduce a mixture distribution over $\{\boldsymbol{z}_0^i\}_{i=1}^L$, constructed by diagonal Gaussian distribution $\mathcal{N}$

$$q_\phi(\boldsymbol{z}_0^i \mid X) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{z}_0^i}, \boldsymbol{\sigma}_{\boldsymbol{z}_0^i}) \tag{7}$$

$$q_\phi(\boldsymbol{z}_0 \mid X) = \sum_{i=1}^{L} \pi_i * q_\phi(\boldsymbol{z}_0^i \mid X), \tag{8}$$

where $\boldsymbol{\mu}_{\boldsymbol{z}_0^i} = h(\boldsymbol{z}_0^i)$, and $\boldsymbol{\sigma}_{\boldsymbol{z}_0^i} = \mathrm{Softplus}(h(\boldsymbol{z}_0^i))$. h denotes a feed-forward neural network and Softplus is the activation function. π denote the mixing coefficients for the L components. The entire operation is summarized by lines 3–5 in Algorithm 1. More details about π will be discussed below in the section about supervised learning.

Evolving Forward in Time In this part, we first draw an instance from the posterior distribution $q_\phi(\boldsymbol{z}_0 \mid X)$ to obtain $\boldsymbol{z}_0$ (line 6 in Algorithm 1), which will further be used as a representation of the time series sample and also as the initial point of extrapolation. We then start from $\boldsymbol{z}_0$ and propagate the latent state $\boldsymbol{z}$ forward along the timeline, with $\Delta t_i = t_i - t_0$ (line 7). Thus, $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_L$ can be calculated for all the L timestamps by another call of the IVP solver (line 8):

$$\{\boldsymbol{z}_i\}_{i=1}^L = \{\mathrm{IVPSolve}\,(\boldsymbol{z}_0, \Delta t_i)\}_{i=1}^L. \tag{9}$$

The multivariate observation $\boldsymbol{x}_i$ can then be obtained from $\boldsymbol{z}_i$ using the data reconstruction module explained previously (line 9 in Algorithm 1). The entire operation is mathematically represented as approximating $p_\theta(X \mid z_0)$.

In terms of capturing temporal dependencies, RNN-related models repeatedly operate on sequential observations and extract useful information in an autoregressive way. Using IVP-VAE, the dependence is captured by Neural ODEs with derivatives, and Neural Flows with invertible transformations. Thus, the encoder and decoder do not require any recurrent operation, as all latent states at different time points can evolve independently given an ODE.

Invertibility and Bidirectional Evolving The mechanism that one neural IVP solver works for both the encoder and decoder by solving IVPs in opposite time directions is achieved by utilizing the invertibility of IVP solvers. A detailed introduction to neural IVP solvers and the invertibility phenomena can be found in Appendix A.1.

Training

The IVP-VAE model can be trained both on unsupervised and supervised learning.

Unsupervised Learning To learn the parameters of our IVP-VAE model given a dataset of sparse and irregularly sampled time series, we define the learning objective for one sample X as

$$\begin{aligned}\mathcal{L}_{\mathrm{VAE}}(\phi,\theta) =& \mathbb{E}_{\boldsymbol{z}_0 \sim q_\phi(\boldsymbol{z}_0|X)}\left[\log p_\theta\left(X \mid \boldsymbol{z}_0\right)\right] \\ & - \frac{1}{L}\sum_{i=1}^{L} D_{KL}(q_\phi(\boldsymbol{z}_0^i \mid X) \| p(\boldsymbol{z}_0)),\end{aligned} \tag{10}$$

which corresponds to the evidence lower bound (ELBO) (Kingma and Welling 2014).

As mentioned earlier, each $\boldsymbol{z}_0^i$ should approximate $\boldsymbol{z}_0$ during training, so the second term of $\mathcal{L}_{\mathrm{VAE}}(\phi,\theta)$ is the average of KL-divergence loss between $\left\{q_\phi(\boldsymbol{z}_0^i \mid X)\right\}_{i=1}^L$ and $p(\boldsymbol{z}_0)$. Given that not all data dimensions are observed at all time points, we calculate the reconstruction loss based on all available observations.

Supervised Learning **Forecasting.** The model's capability of extrapolation can be used for time series forecasting. To produce value predictions out of the input time window T, one can simply continue to propagate the latent state $\boldsymbol{z}_i$ using the same neural IVP solver to any desired time points, e.g. in the forecast time window $[T, T+\tau]$, without adding any additional component. After propagation, the same reconstruction module can be used to map $\boldsymbol{z}_i$ to $\hat{\boldsymbol{x}}_i$, thus obtaining $\hat{X}^\tau$, which is the forecasted content with regard to the truth X^τ. We combine $\mathcal{L}_{\mathrm{VAE}}$ with the reconstruction error $\mathcal{L}_{\mathrm{Re}}$ on X^τ to obtain Equation 11, where α is a hyperparameter.

$$\mathcal{L}_{\mathrm{Forecast}}(\phi,\theta) = \mathcal{L}_{\mathrm{VAE}}(\phi,\theta) + \alpha \cdot \mathcal{L}_{\mathrm{Re}}(\hat{X}^\tau \| X^\tau) \quad (11)$$

Classification. We can also augment IVP-VAE with a classifier that leverages the latent state evolving as feature extraction and representation learning. We define this portable classification component to be of the form $p_\lambda(y \mid \boldsymbol{z}_0)$, where λ represents model parameters (essentially a feed-forward network). This leads to an augmented learning objective, as shown in Equation 12, where CE is the cross entropy loss.

$$\begin{aligned}\mathcal{L}_{\mathrm{Class}}(\phi,\theta,\lambda) =& \mathcal{L}_{\mathrm{VAE}}(\phi,\theta) \\ &+ \alpha \cdot \mathrm{CE}(p(y) \| p_\lambda(y \mid \boldsymbol{z}_0))\end{aligned} \quad (12)$$

The value of the mixing coefficient π_i in Equation 8 depends on the performed task. There exist various methods to determine the mixing coefficients in a mixture distribution. In our proposed model, we empirically obtained two different settings for the mixing coefficients: $\pi_i = \frac{1}{L}$ for classification and $\pi_i = \frac{D_{KL}(q_\phi(\boldsymbol{z}_0^i|X)|p(\boldsymbol{z}_0))}{\sum_{j=1}^{L} D_{KL}(q_\phi(\boldsymbol{z}_0^j|X)|p(\boldsymbol{z}_0))}$ for forecasting tasks.

Experiments

In this section, we present the experimental protocol and the range of baseline models used along with the EHR datasets.

Datasets

We evaluate our model on three real-world public EHR datasets from the PhysioNet platform (Goldberger et al. 2000): MIMIC-IV (Johnson et al. 2020, 2023), PhysioNet 2012 (Silva et al. 2012) and eICU (Pollard et al. 2018, 2019).

The **MIMIC-IV** dataset is a multivariate EHR time series dataset consisting of sparse and irregularly sampled physiological signals collected at Beth Israel Deaconess Medical Center from 2008 to 2019. After data preprocessing following a similar procedure to Biloš et al. (2021), 96 variables covering patient in- and outputs, laboratory measurements, and prescribed medications, are extracted over the first 48 hours after ICU admission. We obtain 26,070 records and use them for both forecasting and classification.

The **PhysioNet 2012** dataset was published as part of the PhysioNet/Computing in Cardiology Challenge 2012 with the objective of in-hospital mortality prediction. It includes vital signs, laboratory results, and demographics of patients admitted to an ICU. We use the provided 4,000 admissions from the challenge training set and 37 features over the first 48 hours after patient admission following Biloš et al. (2021).

The **eICU** Collaborative Research Database is a multi-center dataset of patients admitted to ICUs at 208 hospitals located throughout the United States between 2014 and 2015. We follow the preprocessing procedure presented in Romero et al. (2022) and extract 14 features over the first 48 hours after ICU admission for 12,312 admissions.

	MIMIC-IV	PhysioNet 2012	eICU
# Samples	26,070	3,989	12,312
# Variables	96	37	14
Missing rate (%)	97.95	84.34	65.25
Average length	173.4	75.0	114.55
Positive rate (%)	13.39	13.89	17.61
Granularity	1 min	1 min	1 min

Table 1: Key information of the three datasets after preprocessing: Number of admissions used, number of selected variables, overall percentage of missing values, average sequence length over admissions, positive rate for mortality, granularity of measurements.

The key information of the three datasets after preprocessing is summarized in Table 1. MIMIC-IV has the highest rate of missing values, the longest average sequence length, and the smallest positive rate for mortality. The eICU data is the least sparse, with a missing rate of only about 65 %. The full list of selected variables of each dataset can be found in Appendix A.3.

Baselines

We compare our model against several baselines for the forecasting and classification of multivariate irregular time-series.

- **GRU-Δ_t** concatenates feature values with masking variable and time interval Δ_t as input (Rubanova, Chen, and Duvenaud 2019).
- **GRU-D** incorporates missing patterns using GRU combined with a learnable decay mechanism on both the input sequence and hidden states (Che et al. 2018).
- **mTAN** leverages an attention mechanism to learn temporal similarity and time embeddings (Shukla and Marlin 2021a).
- **GRU-ODE-Bayes** couples continuous-time ODE dynamics with discrete Bayesian update steps (De Brouwer et al. 2019).
- **CRU** constructs continuous recurrent cells using linear stochastic differential equations and Kalman filters (Schirmer et al. 2022).
- **Raindrop** represents dependencies among multivariates with a graph whose connectivity is learned from time series (Zhang et al. 2022).
- **Latent-ODE** uses an ODE-RNN encoder and Neural ODE decoder in a VAE architecture (Rubanova, Chen, and Duvenaud 2019).
- **Latent-Flow** replaces the ODE component of Latent-ODE with more efficient Neural Flow models (Biloš et al. 2021).

Corresponding to two Latent-based models, we evaluate IVP-VAE with two types of IVP solvers, i.e. one with ODE called IVP-VAE-ODE and another with Flow called IVP-VAE-Flow. Hyperparameter settings are described in Appendix A.2. Latent-ODE and Latent-Flow, which are the primary baselines for our model, are jointly referred to as *Latent-based* models below.

Experimental Protocols

All three datasets are used for forecasting and classification experiments. Each dataset is randomly split into 80% for training, 10% for validation and 10% for testing. Following previous works (Rubanova, Chen, and Duvenaud 2019; Shukla and Marlin 2021a; Zhang et al. 2022), we repeat each experiment five times using different random seeds to split datasets and initialize model parameters.

In forecasting experiments, we use the first 24 hours of data as input and prediction the next 24 hours of data. We assess models' performance using the mean squared error (MSE). For classification experiments, we focus on predicting in-hospital mortality using the first 24 hours of data. Due to class imbalance in these datasets, we assess classification performance using area under the ROC curve (AUROC) and area under the precision-recall curve (AUPRC). To compare the models' running speed, we also report T-epoch (Biloš et al. 2021; Horn et al. 2020; Shukla and Marlin 2021a), which is the time that each model needs to complete one epoch (counted in seconds). All models were tested in the same computing environment with NVIDIA Tesla V100 GPUs.

Considering the fact that even though some public EHR datasets have sufficient general samples for training complex deep learning models, when it comes to a specific group of patients or a specific medical phenomenon, the available data for training are usually not sufficient (Shickel et al. 2017). We also deploy our model and other baselines in experiments with limited samples, conducting a comprehensive comparison across various dataset sizes. The samples are drawn from the MIMIC IV dataset. The test dataset consistently contains 2,000 samples, whereas the number of samples for training and validation ranges from 250 to 4,000. These small-sized datasets are then divided into training and validation sets at a 4:1 ratio. Both classification and forecasting tasks are conducted within this setting.

		ODE		Flow	
		IVP-VAE	Latent	IVP-VAE	Latent
MIMIC-IV					
Classification	AUROC	**0.802**	0.768	**0.805**	0.786
	AUPRC	**0.422**	0.393	**0.427**	0.404
	T-forward	**0.066**	2.536	**0.017**	0.784
	T-epoch	**1478.8**	5270.3	**1445.8**	3105.5
	# Epochs	**12.6**	56.2	**10.8**	57.8
	# Parameters	209,677	**199,017**	**325,017**	429,697
Forecasting	MSE	**0.724**	0.769	**0.727**	0.755
	T-forward	**0.106**	3.059	**0.025**	1.063
	T-epoch	**155.4**	4294.9	**81.5**	2272.0
	# Epochs	**31.8**	37.2	**35.6**	42.8
	# Parameters	112,776	**102,116**	**228,116**	332,796
PhysioNet 2012					
Classification	AUROC	**0.770**	0.767	**0.771**	0.766
	AUPRC	0.359	**0.364**	**0.362**	0.327
	T-forward	**0.031**	1.072	**0.009**	0.285
	T-epoch	**35.6**	333.4	**32.6**	166.5
	# Epochs	**19.6**	89.2	**19.4**	96.2
	# Parameters	**174,218**	188,338	**289,558**	419,018
Forecasting	MSE	**0.563**	0.586	**0.567**	0.584
	T-forward	**0.072**	0.916	**0.012**	0.307
	T-epoch	**20.2**	292.9	**8.2**	264.7
	# Epochs	54.4	**38.4**	68.0	**44.2**
	# Parameters	**77,317**	91,437	**192,657**	322,117
eICU					
Classification	AUROC	**0.786**	0.783	**0.786**	0.781
	AUPRC	0.468	**0.477**	0.472	**0.482**
	T-forward	**0.033**	2.733	**0.009**	0.539
	T-epoch	**342.5**	2296.1	**319.4**	1127.4
	# Epochs	**16.0**	78.8	**23.0**	93.0
	# Parameters	**160,395**	184,175	**275,735**	414,855
Forecasting	MSE	**0.596**	0.598	**0.581**	0.594
	T-forward	**0.081**	2.604	**0.012**	0.655
	T-epoch	**77.7**	1778.3	**28.2**	616.3
	# Epochs	60.2	**32.0**	78.2	**42.5**
	# Parameters	**160,395**	184,175	**275,735**	414,855

Table 2: Detailed comparison of IVP-VAE and its predecessor using different IVP solvers on three datasets for classification and forecasting. We compare the time needed for one forward pass (T-forward) and for one epoch (T-epoch), number of epochs, and number of parameters. Better results are in bold.

Results and Analyses

In this section, we evaluate IVP-VAE's capability of data modeling and representation learning for EHR time series data. There are different branches of methods for irregular time series. We first make a thorough comparison of our designs and their Latent-based predecessors to show the improvement. Afterward, we compare our designs against the state-of-art and representative methods from other branches.

Improvements Over Latent-Based Models

To have a clear view of the improvement of performance and efficiency, we make a detailed comparison of our design and Latent-based models in Table 2. Regarding performance in classification (AUROC & AUPRC, larger means better) and forecasting (MSE, smaller means better) tasks, IVP-VAE generally outperforms Latent-based models across all datasets.

We further compare efficiency of these models in terms of T-epoch and T-forward (the time taken by each model to complete one forward run). Clearly, IVP-VAE models are able to achieve a significant speed advantage over the corresponding Latent-based models. For instance, on forecasting tasks of MIMIC-IV, IVP-VAE-Flow is about 42 times faster than Latent-Flow in terms of T-forward. Since T-epoch includes T-forward as well as the time for data loading, loss calculation, backpropagation, etc., which significantly contribute to the computation time, the improvement in T-epoch is not as significant as in T-forward. Nevertheless, IVP-VAE-Flow is still more than 28 times faster than Latent-Flow. The speed advantage is achieved by eliminating recurrent operations and solving IVPs in parallel.

Furthermore, we compare IVP-VAE with its counterparts on convergence rate. As indicated by # Epochs, IVP-VAE models converge significantly faster than Latent-based models, with IVP-VAE models needing lesser epochs to achieve

	MIMIC-IV			PhysioNet 2012			eICU		
	MSE	AUROC	AUPRC	MSE	AUROC	AUPRC	MSE	AUROC	AUPRC
GRU-Δ_t	0.730±0.014	**80.9±0.6**	42.0±2.0	0.587±0.055	72.0±4.4	29.0±4.5	0.583±0.009	76.1±1.4	42.8±2.1
GRU-D	0.736±0.005	78.6±0.9	41.9±1.3	0.588±0.060	76.2±3.2	32.9±4.3	0.578±0.007	**79.6±1.5**	**47.7±2.4**
mTAN	0.715±0.011	76.6±0.6	37.9±2.4	0.588±0.050	76.2±2.8	33.8±5.3	0.582±0.010	76.9±2.4	45.1±3.2
Raindrop	-	77.1±1.4	36.8±2.8	-	75.3±2.3	30.9±3.9	-	76.6±2.1	45.1±2.7
GOB	0.809±0.014	-	-	0.619±0.029	-	-	0.664±0.012	-	-
CRU	0.946±0.016	-	-	0.688±0.032	-	-	0.820±0.044	-	-
IVP-VAE-Flow	**0.727±0.013**	80.5±0.5	**42.7±1.4**	**0.567±0.038**	**77.1±3.0**	**36.2±5.3**	**0.581±0.009**	78.6±1.7	47.2±0.029

Table 3: Comparison of the proposed IVP-VAE-Flow model and state-of-the-art baselines. '-' denotes that a model doesn't support the task. We report test MSE for forecasting and AUROC $\left(\times 10^{-2}\right)$ and AUPRC $\left(\times 10^{-2}\right)$ for mortality prediction on three datasets. IVP-VAE-Flow achieves competitive performance across all datasets and tasks.

the best validation accuracy. This advantage is achieved by the parameter sharing mechanism in our models, i.e. one IVP solver for both the encoder and decoder. Multiplying the time per epoch by the number of epochs to obtain the total training time, we find that IVP-VAE based models are at least one order of magnitude faster than Latent-based models for both classification and forecasting tasks.

Regarding model size (# Parameters in Table 2), IVP-VAE-Flow is smaller than Latent-Flow in all scenarios. IVP-VAE-ODE is smaller than Latent-ODE in most cases, except for forecasting tasks on MIMIC-IV. Compared to Latent-based models, IVP-VAE (1) eliminates recurrent units, (2) uses one IVP solver instead of two, and (3) adds in the embedding and reconstruction modules. Factor (1) and (2) can reduce the number of parameters, while factor (3) increases the number of parameters. The overall parameter difference is the result of the superposition of these three factors.

Comparison Against Other Representative Models

We compare IVP-VAE-Flow as the best-performing proposed model with other state-of-the-art and representative baselines. The results of forecasting (in MSE) and classification (AUROC, AUPRC) experiments on the three datasets are presented in Table 3. For each metric, we use bold font to indicate the best result. When compared with other state-of-the-art baseline models, the IVP-VAE-Flow model consistently achieves at least the second-best result. Also, IVP-VAE even achieves the best results for the PhysioNet 2012 dataset for both forecasting and classification. Overall, the proposed method exhibits competitive performance across all the datasets and tasks.

Experiments on Small Datasets

To further demonstrate the capabilities of the proposed model, we examine the performance under low sample size conditions. This scenario is analogous to a rare disease setting in the field of EHR prediction, where data can only be obtained for a small cohort of patients. In such cases, the effectiveness of models in capturing temporal evolving patterns and rapidly updating parameters becomes essential. Figure 2 compares the performance of 4 typical methods on small datasets where we collected only a limited number of samples for model training and validation. As we can see, for both forecast and classification tasks, IVP-VAE based model consistently and substantially outperforms other approaches across all settings with different sample sizes. The advantage of the model on small datasets is also due to its parameter sharing mechanism in the encoder and decoder.

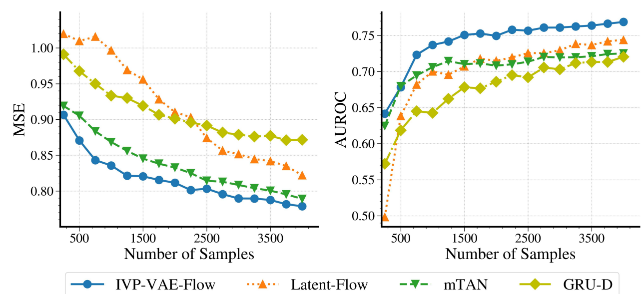

Figure 2: Performance comparison on small datasets: (Left) MSE for forecasting and (right) AUROC for classification task. IVP-VAE based models consistently and substantially outperform all baseline approaches across all datasets with different number of samples.

Conclusion and Discussion

In this paper, we have presented a faster and lighter continuous-time generative model IVP-VAE, which is able to model and learn representations of irregular sampled EHR time series by purely solving IVPs in parallel under the VAE architecture. Our results showed that the proposed models perform comparable or better than other baselines on classification and forecasting tasks, while offering training times that are one order of magnitude faster than previous continuous-time methods. Further experiments on small datasets showed that our model has an advantage in scenarios where the number of training samples is limited. Based on this, more work can be done to demonstrate the ability of IVP-VAE to model irregular sample time series with diverse datasets, not only EHR datasets, and different tasks like missing value imputation, time series regression, etc.

Acknowledgments

This research was funded by the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor with grant No. 01DD20003.

References

Abul-Husn, N. S.; and Kenny, E. E. 2019. Personalized Medicine and the Power of Electronic Health Records. *Cell*, 177(1): 58–69.

Biloš, M.; Sommer, J.; Rangapuram, S. S.; Januschowski, T.; and Günnemann, S. 2021. Neural Flows: Efficient Alternative to Neural ODEs. In *Advances in Neural Information Processing Systems*, volume 34, 21325–21337. Curran Associates, Inc.

Cao, W.; Wang, D.; Li, J.; Zhou, H.; Li, L.; and Li, Y. 2018. BRITS: Bidirectional Recurrent Imputation for Time Series. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, 8: 1–12.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Chien, J.-T.; and Chen, Y.-H. 2021. Continuous-time Attention for Sequential Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7116–7124.

De Brouwer, E.; Simm, J.; Arany, A.; and Moreau, Y. 2019. GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Fey, M.; Lenssen, J. E.; Weichert, F.; and Müller, H. 2018. SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 869–877.

Gao, Y.; Cai, G.-Y.; Fang, W.; Li, H.-Y.; Wang, S.-Y.; Chen, L.; Yu, Y.; Liu, D.; Xu, S.; Cui, P.-F.; et al. 2020. Machine Learning Based Early Warning System Enables Accurate Mortality Risk Prediction for COVID-19. *Nature Communications*, 11(1): 5033.

Goldberger, A. L.; Amaral, L. A. N.; Glass, L.; Hausdorff, J. M.; Ivanov, P. C.; Mark, R. G.; Mietus, J. E.; Moody, G. B.; Peng, C.-K.; and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101.

Harutyunyan, H.; Khachatrian, H.; Kale, D. C.; Ver Steeg, G.; and Galstyan, A. 2019. Multitask Learning and Benchmarking with Clinical Time Series Data. *Scientific Data*, 6(1): 96.

Horn, M.; Moor, M.; Bock, C.; Rieck, B.; and Borgwardt, K. 2020. Set Functions for Time Series. In *International Conference on Machine Learning*, volume 119, 4353–4363. PMLR.

Johnson, A. E. W.; Bulgarelli, L.; Pollard, T. J.; Horng, S.; Celi, L. A.; and Mark, R. G. 2020. MIMIC-IV (version 1.0). https://doi.org/10.13026/s6n6-xd98. Accessed: 2021-12-05.

Johnson, A. E. W.; Bulgarelli, L.; Shen, L.; Gayles, A.; Shammout, A.; Horng, S.; Pollard, T. J.; Moody, B.; Gow, B.; Lehman, L.-w. H.; Celi, L. A.; and Mark, R. G. 2023. MIMIC-IV, A Freely Accessible Electronic Health Record Dataset. *Scientific Data*, 10(1): 1–9.

Johnson, A. E. W.; Pollard, T. J.; Shen, L.; wei H. Lehman, L.; Feng, M.; Ghassemi, M.; Moody, B.; Szolovits, P.; Celi, L. A.; and Mark, R. G. 2016. MIMIC-III, A Freely Accessible Critical Care Database. *Scientific Data*, 3: 160035.

Kim, Y. J.; and Chi, M. 2018. Temporal Belief Memory: Imputing Missing Data During RNN Training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2326–2332.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*.

Li, Q.; and Xu, Y. 2019. VS-GRU: A Variable Sensitive Gated Recurrent Neural Network for Multivariate Time Series with Massive Missing Values. *Applied Sciences*, 9(15).

Li, S. C.-X.; and Marlin, B. M. 2020. Learning from Irregularly-sampled Time Series: A Missing Data Perspective. In *International Conference on Machine Learning*, 5937–5946. PMLR.

Lipton, Z. C.; Berkowitz, J.; and Elkan, C. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv:1506.00019.

McDermott, M.; Nestor, B.; Kim, E.; Zhang, W.; Goldenberg, A.; Szolovits, P.; and Ghassemi, M. 2021. A Comprehensive EHR Timeseries Pre-training Benchmark. In *Proceedings of the Conference on Health, Inference, and Learning*, 257–278.

Pollard, T. J.; Johnson, A. E. W.; Raffa, J. D.; Celi, L. A.; Badawi, O.; and Mark, R. 2019. eICU Collaborative Research Database (version 2.0). https://doi.org/10.13026/C2WM1R. Accessed: 2022-10-01.

Pollard, T. J.; Johnson, A. E. W.; Raffa, J. D.; Celi, L. A.; Mark, R. G.; and Badawi, O. 2018. The eICU Collaborative Research Database, A Freely Available Multi-center Database for Critical Care Research. *Scientific Data*, 5(1): 1–13.

Purushotham, S.; Meng, C.; Che, Z.; and Liu, Y. 2018. Benchmarking Deep Learning Models on Large Healthcare Datasets. *Journal of Biomedical Informatics*, 83: 112–134.

Romero, D. W.; Kuzina, A.; Bekkers, E. J.; Tomczak, J. M.; and Hoogendoorn, M. 2022. CKConv: Continuous Kernel Convolution For Sequential Data. In *International Conference on Learning Representations (ICLR)*.

Rubanova, Y.; Chen, R. T. Q.; and Duvenaud, D. K. 2019. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Schirmer, M.; Eltayeb, M.; Lessmann, S.; and Rudolph, M. 2022. Modeling Irregular Time Series with Continuous Recurrent Units. In *International Conference on Machine Learning (ICML)*, 19388–19405. PMLR.

Shickel, B.; Tighe, P. J.; Bihorac, A.; and Rashidi, P. 2017. Deep EHR: A Survey of Recent Advances in Deep Learning

Techniques for Electronic Health Record (EHR) analysis. *IEEE Journal of Biomedical and Health Informatics*, 22(5): 1589–1604.

Shukla, S. N.; and Marlin, B. M. 2021a. Multi-Time Attention Networks for Irregularly Sampled Time Series. In *International Conference on Learning Representations (ICLR)*.

Shukla, S. N.; and Marlin, B. M. 2021b. A Survey on Principles, Models and Methods for Learning from Irregularly Sampled Time Series. arXiv:2012.00168.

Silva, I.; Moody, G.; Scott, D. J.; Celi, L. A.; and Mark, R. G. 2012. Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012. In *Computing in Cardiology*, volume 39, 245–248.

Sun, C.; Hong, S.; Song, M.; and Li, H. 2020. A Review of Deep Learning Methods for Irregularly Sampled Medical Time Series Data. arXiv:2010.12493.

Syed, M.; Syed, S.; Sexton, K.; Syeda, H. B.; Garza, M.; Zozus, M.; Syed, F.; Begum, S.; Syed, A. U.; Sanford, J.; and Prior, F. 2021. Application of Machine Learning in Intensive Care Unit (ICU) Settings Using MIMIC Dataset: Systematic Review. *Informatics*, 8(1): 16.

Tipirneni, S.; and Reddy, C. K. 2022. Self-Supervised Transformer for Sparse and Irregularly Sampled Multivariate Clinical Time-Series. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(6): 1–17.

Weerakody, P. B.; Wong, K. W.; Wang, G.; and Ela, W. 2021. A Review of Irregular Time Series Data Handling with Gated Recurrent Neural Networks. *Neurocomputing*, 441: 161–178.

Zhang, X.; Zeman, M.; Tsiligkaridis, T.; and Zitnik, M. 2022. Graph-Guided Network for Irregularly Sampled Multivariate Time Series. In *International Conference on Learning Representations (ICLR)*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-series Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11106–11115.