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

Jingge Xiao[1], Leonie Basso[2], Niloy Ganguly[3] and Sandipan Sikdar[4]
[1 2 4]L3S Research Center, Leibniz Universität Hannover
[3]Indian Institute of Technology Kharagpur
[1]xiao,[2]basso,[4]sandipan.sikdar@l3s.de
[3]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 based on its invertibility, which leads to fewer parameters and faster convergence. Experiments on two EHR datasets show that the proposed approach achieves comparable classification performance while gaining more than *10x* speedup over other continuous-time counterparts.

1 Introduction

Irregularly sampled time series are ubiquitous in a variety of domains, including healthcare. Electronic Health Record (EHR) data contains time series of patient information, such as vital signs and laboratory results, which can have missing values and unequal time intervals between successive measurements. The irregularity is caused mainly due to unstructured manual processes, event-driven recordings, device failure, and also different sampling frequencies among multiple variables (Weerakody et al., 2021)). These complexities make learning and modeling clinical time series data particularly challenging for classical machine learning models (Shukla & Marlin, 2020).

Significant progress has been made towards developing models that can handle irregularly sampled time series data (Shukla & Marlin, 2020). Recent models can be categorized as recurrent neural network (RNN)-based (Kim & Chi, 2018; Cao et al., 2018; Li & Xu, 2019), convolutional neural network (CNN)-based (Romero et al., 2022; Li & Marlin, 2020; Fey et al., 2018) or attention-based (Shukla & Marlin, 2021; Zerveas et al., 2021; Tipirneni & Reddy, 2022). Even though some of these models allow for incorporating continuous time information, they are essentially discrete. Neural ODEs (Chen et al., 2018) are a continuous-time model based on ordinary differential equations (ODE) that can naturally handle irregularly sampled data. To address the irregularity in time series, Rubanova et al. (2019) develop Latent-ODE by integrating Neural ODEs and RNN into a variational autoencoder (VAE) (Kingma & Welling, 2013). Biloš et al. (2021) propose an efficient alternative by directly modeling the solution of ODEs with a neural network, obtaining a Flow-based variant noted as Latent-Flow in this paper. However, when analyzing time series, these continuous-time models require sequential processing of data which makes them inefficient and hard to train.

In this work, we propose a continuous-time generative model, IVP-VAE, for irregularly sampled time series. Unlike Latent-ODE and Latent-Flow, our model takes variational approximation purely as solving initial value problems (IVPs). States at different observation times are mapped to the latent variable z_0 by solving different IVPs in parallel, thereby achieving significant speedup over existing continuous-time models. As IVP-solvers are inherently invertible, by solving IVPs in opposite time directions, we fuse the encoding and decoding module as the same solver, which results in smaller model size and faster convergence. We validate our model across two EHR time series datasets for the task of mortality prediction, where IVP-VAE's advantage is clearly demonstrated.

2 Methodology

2.1 Problem Formulation

Given an irregular time series database $\mathcal{X} = \{X_1, \dots, X_N\}$ collected within a fixed time window T, where N is the total number of samples. The n-th sample X_n consists of a sequence of observations $X_n = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^{L}$ where L is the sequence length, i.e. the number of observations, and each $\boldsymbol{x}_i$ represents the observation at time t_i. Specifically, $\boldsymbol{x}_i \in \mathbb{R}^D$ and D is the number of variables recorded, thus we have $X_n \in \mathbb{R}^{L\times D}$. L's value varies among different samples because of the irregularity. Our goal is to first build a generative model for irregular time series, and then concatenate a classifier to make predictions.

2.2 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 continuous process F and models it following an ordinary differential equation $\frac{dF(t)}{dt} = f(t, \boldsymbol{z}_t)$, where $\boldsymbol{z}_t$ is the hidden state of $\boldsymbol{x}_t$. Given an initial condition $(t_0, \boldsymbol{z}_0)$, the value of $\boldsymbol{z}$ at t_i could be computed by numerical methods, including Neural ODEs (Chen et al., 2018) and Neural Flows (Biloš et al., 2021). Neural ODEs parameterize f with uniformly Lipschitz continuous neural networks. Here f specifies the derivative at every $t \in [t_0, T]$. States of the continuous process are calculated by numerical integrators. Neural Flows propose to directly model the solution curve F with invertible neural networks. As they all propagate hidden states by solving initial value problems, we collectively refer to them as neural IVP-solvers in this paper.

As for modeling irregular time series, Latent-ODE (Rubanova et al., 2019) is built within the VAE architecture with an ODE-RNN hybrid as the encoder and an ODE-version IVP-solver as decoder. Given a hidden state $\boldsymbol{z}_{i-1}$, to obtain the next hidden state, they evolve $\boldsymbol{z}$ continuously until the next observation $\boldsymbol{x}_i$ at time t_i and then update the state with an RNN cell, as expressed by $\boldsymbol{z}_i^- = \mathrm{IVPSolve}((\boldsymbol{z}_{i-1}, t_{i-1}), t_i)$ and $\boldsymbol{z}_i = \mathrm{RNN}(x_t, \boldsymbol{z}_i^-)$. Latent-Flow can be obtained by replacing the ODE module with Flows (Biloš et al., 2021). Both Latent-ODE and Latent-Flow were proven to be effective in modeling irregular time series. However, sequentially solving IVPs makes these models computationally less efficient.

2.3 Proposed model IVP-VAE

Our proposed model IVP-VAE is based on two design principles - (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 encoding and decoding. 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.

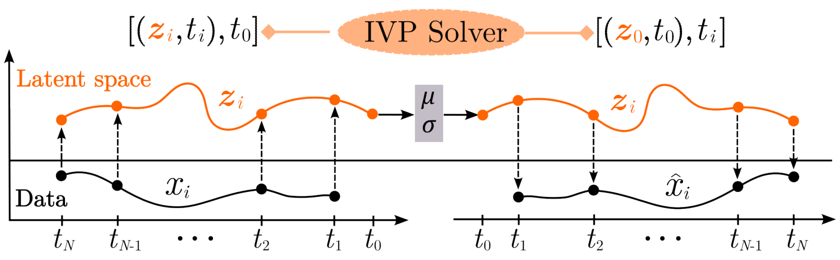
$[(\boldsymbol{z}_i, t_i), t_0]$
IVP Solver
$[(\boldsymbol{z}_0, t_0), t_i]$
Latent space
$\boldsymbol{z}_i$
μ
σ
$\boldsymbol{z}_i$
Data
x_i
$\hat{x}_i$
t_N t_{N-1} ··· t_2 t_1 t_0
t_0 t_1 t_2 ··· t_{N-1} t_N

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 timeline to obtain state $\boldsymbol{z}_i$ using the same IVP solver. A reconstruction module then maps $\boldsymbol{z}_i$ back to data $\hat{\boldsymbol{x}}_i$.

Embedding and reconstruction: Given a time series $X = \{(\boldsymbol{x}_i, t_i)\}_{i=0}^{L}$, we first generate corresponding masks $\{\boldsymbol{m}_i\}_{i=0}^{L}$ indicating 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 $\boldsymbol{v}_i$ to produce $\boldsymbol{z}_i$ which represents the state of continuous process at t_i. On the decoder side, we design a similar module for reconstruction that maps $\boldsymbol{z}_i$ to $\boldsymbol{x}_i$. The aim of adding embedding and reconstruction modules is to create a latent space in which state $\boldsymbol{z}$ evolves.

Evolving backward in time: The overall goal here is to approximate the posterior, i.e., learn an approximate distribution to sample $\boldsymbol{z}_0$. For $\boldsymbol{x}_i$, the initial condition is defined as $(\boldsymbol{z}_i, t_i)$ in the encoder. The task of the Neural IVP-solver is to start from t_i, move towards t_0 continuously and calculate $\boldsymbol{z}_0$:

$$\boldsymbol{z}_0^i = \text{IVPSolve}(\boldsymbol{z}_i, \Delta t_i), \tag{1}$$

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, we obtain $\{\boldsymbol{z}_0^i\}_{i=0}^{L}$ for all $\{\boldsymbol{x}_i, \Delta t_i\}_{i=0}^{L}$. Then an average integration is applied on $\{\boldsymbol{z}_0^i\}_{i=0}^{L}$ to produce $\boldsymbol{z}_0$. Finally, we construct a distribution over the latent variable using a diagonal Gaussian distribution with mean and variance given by the output of fully connected layers g.

$$q_\phi(\boldsymbol{z}_0 \mid X) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{z}_0}, \boldsymbol{\sigma}_{\boldsymbol{z}_0}), \text{ where } \boldsymbol{\mu}_{\boldsymbol{z}_0} = g(\boldsymbol{z}_0), \boldsymbol{\sigma}_{\boldsymbol{z}_0} = \text{Softplus}(g(\boldsymbol{z}_0)), \tag{2}$$

where we define $q_\phi(\boldsymbol{z}_0 \mid X)$ to be the distribution over latent variable $\boldsymbol{z}_0$ induced by time series X.

Note that the encoder does not require any recurrent operation, as all the latent states evolve directly to $\boldsymbol{z}_0$ which resolves the sequential computation bottleneck of the existing continuous-time models.

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 $\boldsymbol{z}_0$, which could further be used as a representation of the time series sample. Afterward, we start from $\boldsymbol{z}_0$ and propagate latent state $\boldsymbol{z}$ forward along the timeline, with $\Delta t_i = t_i - t_0$. So that $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_L$ for all the L timestamps can be calculated by another call of the IVP-solver:

$$\{\boldsymbol{z}_i\}_{i=0}^{L} = \text{IVPSolve}\left(\{\boldsymbol{z}_0, \Delta t_i\}_{i=1}^{L}\right) \tag{3}$$

X can then be generated from $\boldsymbol{z}_0$ using the data reconstruction module explained earlier in this section. The entire operation is mathematically represented as $p_\theta(X \mid \boldsymbol{z}_0)$.

Within IVP-VAE, there is only one IVP-solver working for both encoder and decoder with opposite Δt. This can be achieved due to the invertibility of neural IVP-solvers. For Neural ODEs, state $\boldsymbol{z}$ at different timestamps can be computed following $\boldsymbol{z}_i = \boldsymbol{z}_{i-1} + \int_{t_{i-1}}^{t_i} f(t, \boldsymbol{z}_t)dt$ where f is determined within its domain, while Δt can be negative or positive. If positive, the state evolves forward in time. If negative, it evolves backward in time. Similarly, a flow ξ is mathematically defined as a group action on a set $Z, \xi : Z \times \mathbb{R} \to Z$, where for all $z \in Z$ and real number t, $\xi(\xi(z, t_i), t_j) = \xi(z, t_j + t_i)$ (Bhatia & Szegö, 1970). This characteristic naturally guarantees its invertibility. If $t_j = -t_i$, then $\xi(\xi(z, t_i), t_j) = \xi(z)$ can describe the process of encoder and decoder.

Training: Given the model is trained on a classification task, we augment IVP-VAE with a supervised learning component with the form $p_\lambda(y \mid \boldsymbol{z}_0)$ (essentially a neural network that maps $\boldsymbol{z}_0$ to the class y) and train on the joint objective -

$$\mathcal{L}_{\text{Class}}(\phi, \theta, \lambda) = \mathcal{L}_{\text{VAE}}(\phi, \theta) + \alpha \cdot \text{CE}(p(y) \| p_\lambda(y \mid \boldsymbol{z}_0)) \tag{4}$$

where α is a hyperparameter, CE represents cross entropy loss and $\mathcal{L}_{\text{VAE}}(\phi, \theta)$ is the evidence lower bound (ELBO) represented as -

$$\mathcal{L}_{\text{VAE}}(\phi, \theta) = \sum_{n=1}^{N} (\mathbb{E}_{\boldsymbol{z}_0 \sim q_\phi(\boldsymbol{z}_0|X)} [\log p_\theta(X \mid \boldsymbol{z}_0)] - D_{KL}(q_\phi(\boldsymbol{z}_0 \mid X) \| p(\boldsymbol{z}_0))). \tag{5}$$

3 Experiments

Datasets and Baselines: We evaluate our model on two public EHR datasets from the PhysioNet platform (Goldberger et al., 2000): MIMIC-IV (Johnson et al., 2023) and PhysioNet 2012 (Silva et al., 2012) with the task of mortality prediction. Datasets are processed following (De Brouwer et al., 2019)'s pipeline. For MIMIC-IV, 22,000 admissions are randomly selected and split into 16,000 for training, 4,000 for validation, and 2,000 for testing. PhysioNet 2012 includes vital signs, laboratory results, as well as demographics. We use the provided 4,000 admissions from training set

A and 41 features over the first 48 hours after patient admission. This dataset is randomly split into 80% training, 10% validation and 10% test sets.

We compare our model against several baselines for the classification of multivariate irregular time-series. GRU-Δ_t (Shukla & Marlin, 2021) concatenates feature values with masking variable and time interval Δ_t as input. GRU-D (Che et al., 2018) combined GRU with a temporal decay mechanism to incorporate missing patterns. Latent-ODE (Rubanova et al., 2019) and Latent-Flow (Biloš et al., 2021) use VAE architectures with different IVP-solvers (see section 2.2). Correspondingly, we evaluate IVP-VAE with two type of solvers, i.e. one with ODE called IVP-VAE-ODE and another with Flow called IVP-VAE-Flow. All experiments were run on NVIDIA Tesla V100 GPUs.

Results and Analyses: We evaluate models' performance with AUROC and AUPRC, as well as average time per epoch (T-epoch) and average time per forward run (T-forward) in the same computing environment. T-epoch and T-forward were counted in seconds. As we can see in Table 1, IVP-VAE based models produce comparable results while significantly improved efficiency. For instance, on PhysioNet 2012, IVP-VAE-Flow is 40 times faster than Latent-Flow in terms of T-forward. T-epoch includes T-forward, as well as time for data loading, loss calculation, and backpropagation, which take up a large part of the time. Therefore, the improvement in T-epoch is not as significant as in T-forward, but IVP-VAE-Flow is still more than 4 times faster than Latent-Flow. Also, IVP-VAE-ODE is much more efficient than its counterpart Latent-ODE. This speed advantage is achieved by eliminating recurrent operations and solving IVPs in parallel.

Table 1: Test AUROC and AUPRC (higher is better) for mortality prediction on datasets MIMIC-IV and PhysioNet 2012. We additionally report time per epoch (T-epoch) and average time per forward run (T-forward). IVP-VAE-based models obtain comparable results and are multiple times faster than other baselines, especially their Latent-based counterparts.

	MIMIC-IV				PhysioNet 2012			
	AUROC	AUPRC	T-epoch	T-forward	AUROC	AUPRC	T-epoch	T-forward
GRU-Δ_t	0.808±0.012	0.382±0.035	1324.3	0.073	0.775±0.021	0.453±0.046	111.3	0.039
GRU-D	0.807±0.010	0.366±0.030	1553.8	0.356	0.770±0.022	0.435±0.052	130.6	0.185
Latent-ODE	0.801±0.005	0.365±0.034	4090.2	3.538	0.794±0.023	0.435±0.023	486.3	2.222
Latent-Flow	0.797±0.004	0.383±0.032	2399.7	0.834	0.794±0.020	0.447±0.055	246.2	0.495
IVP-VAE-ODE	0.807±0.006	0.363±0.028	902.8	0.047	0.791±0.025	0.437±0.049	67.5	0.059
IVP-VAE-Flow	0.802±0.007	0.376±0.026	**878.4**	**0.010**	0.797±0.036	0.448±0.048	**53.2**	**0.013**

Next, Table 2 further compares IVP-VAE with its counterparts on model size and convergence rate. It shows that IVP-VAE models are lighter and converge faster, which is achieved mainly by fusing encoder and decoder with one IVP-solver. If multiplying T-epoch from Table 1 with number of epochs from Table 2, we will see that IVP-VAE based models are more than 10 times faster than Latent based models with regard to training time.

Table 2: Number of parameters, average number of epochs that a model needs to reach its best performance, and computation complexity of VAE-based models. L denotes the length of series, S denotes the steps that Neural ODEs need to solve an IVP. $O(1)$ is obtained because observations at all times are computed in parallel and no sequential dependence is involved.

		# Epochs		Complexity	
	# Parameters	MIMIC-IV	PhysioNet 2012	Encoder	Decoder
Latent-ODE	207K	48.8	32.4	$O(SL)$	$O(S)$
Latent-Flow	472K	45.6	31.8	$O(L)$	$O(1)$
IVP-VAE-ODE	170K	22.4	13.0	$O(S)$	$O(S)$
IVP-VAE-Flow	286K	22.2	13.6	$O(1)$	$O(1)$

4 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 representation of irregular time series by purely solving IVPs in parallel under the VAE architecture. Our results show that the proposed models perform as well as other baselines on classification tasks, while offering training times that are one order of magnitude faster than previous continuous-time methods. Based on this, more work can be done on demonstrating IVP-VAE's capability of modeling irregular sample time series with diverse datasets, and different tasks such as value forecasting, time series regression, etc.

REFERENCES

Nam Parshad Bhatia and Giorgio P Szegö. *Stability theory of dynamical systems*, pp. 5–6. Springer, 1970.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural ODEs. *Advances in Neural Information Processing Systems*, 32, 2021.

Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. BRITS: Bidirectional recurrent imputation for time series. *Advances in Neural Information Processing Systems*, 31, 2018.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8:1–12, 2018.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. *Advances in Neural Information Processing Systems*, 32, 2019.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 869–877, 2018.

Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101, 2000.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1–9, 2023.

Yeo-Jin Kim and Min Chi. Temporal belief memory: Imputing missing data during RNN training. In *International Joint Conference on Artificial Intelligence*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Qianting Li and Yong Xu. Vs-gru: A variable sensitive gated recurrent neural network for multivariate time series with massive missing values. *Applied Sciences*, 9(15), 2019.

Steven Cheng-Xian Li and Benjamin M. Marlin. Learning from irregularly-sampled time series: A missing data perspective. In *International Conference on Machine Learning*, pp. 5937–5946. PMLR, 2020.

David W. Romero, Anna Kuzina, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ODEs for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32, 2019.

Satya Narayan Shukla and Benjamin M. Marlin. A survey on principles, models and methods for learning from irregularly sampled time series. *NeurIPS 2020 Workshop: ML Retrospectives, Surveys & Meta-Analyses*, 11 2020.

Satya Narayan Shukla and Benjamin M. Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021.

Ikaro Silva, George Moody, Daniel J. Scott, Leo A. Celi, and Roger G. Mark. Predicting in-hospital mortality of ICU patients: The PhysioNet/Computing in Cardiology Challenge 2012. In *Computing in Cardiology*, volume 39, pp. 245–248, 2012.

Sindhu Tipirneni and Chandan K Reddy. Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series. *ACM Transactions on Knowledge Discovery from Data*, 16(6):1–17, 2022.

Philip B. Weerakody, Kok Wai Wong, Guanjin Wang, and Wendell Ela. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, 441:161–178, 2021.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2114–2124, 2021.