HiGen: Hierarchical Generation of Irregular Multivariate Time Series

Anonymous authors
Paper under double-blind review

Abstract

Irregular multivariate time series are widely observed in real-world systems, where variables are recorded asynchronously and with varying frequencies. Generating such data is useful for data augmentation, data sharing, and downstream task training. Existing methods usually regularize irregular observations into aligned sequences, discretize them into predefined fixed time bins, or model them with a single smooth continuous-time state evolution that struggles to characterize irregular sampling rhythms. These designs change the original sampling process and struggle to preserve realistic temporal dynamics and asynchronous variable interactions. We propose HiGen, a hierarchical generation framework for irregular multivariate time series. HiGen directly models the original sample-point sample and organizes generation at the sample, window, and point levels. This hierarchy is used to model temporal dynamics through real time intervals at window and point level, and to model variable interactions through sample, window, and point levels. In this way, the model preserves both irregular sampling rhythms and asynchronous variable relationships. Experiments on irregular multivariate time series datasets show that HiGen achieves better downstream utility and generation fidelity than state-of-the-art baselines.

1 Introduction

Multivariate time series are common in clinical monitoring, industrial sensing, wearable devices, and intelligent observation systems (Che et al., 2018; Shukla & Marlin, 2019; 2021), supporting monitoring, risk assessment, downstream tasks, and data-driven decisions. In these real-world scenarios, observations are often irregular and asynchronous, with variable-specific sampling frequencies and sparse observations (Rubanova et al., 2019; De Brouwer et al., 2019; Che et al., 2018; Cao et al., 2018). These patterns are part of the data distribution, reflecting system states, observation policies, and event-triggered mechanisms. However, collecting high-quality irregular multivariate time series at scale is difficult due to privacy, acquisition costs, long-term observation, and scarce critical or rare events. This calls for the need of generating structurally synthetic irregular multivariate time series that reflect real-world data characteristics. Such data can serve as shareable substitutes for private or access-restricted data and support augmentation and downstream task training.

Time series generation learns the distribution of real samples to produce synthetic data with similar characteristics (Yoon et al., 2019; Jeon et al., 2022; Naiman et al., 2024b; Karami et al., 2025). For irregular multivariate time series, generation models must preserve not only plausible values, but also sampling rhythms, asynchronous variable occurrences, and sample level observation horizons. However, learning these characteristics becomes particularly difficult when only a small number of real irregular samples are available for training. Many important patterns are then not sufficiently observed, making it difficult for the model to learn realistic sampling rhythms, variable occurrenceps, and rare states (Gonen et al., 2025).

Existing methods for irregular multivariate time series generation can be roughly divided into three categories. *Regular sequence generation* methods (Yoon et al., 2019; Jeon et al., 2022; Naiman et al., 2024b) impute irregular observations into regular time series and train generators on the regularized data, as shown in Fig. 1(a). *Segment-aligned* methods (Karami et al., 2025; Tian et al., 2024) discretize irregular observations into predefined time bins, as shown in Fig. 1(b). Aligning observations within the same segment weakens their exact timestamps and intra-segment ordering. *Continuous-*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

time generation methods (Rubanova et al., 2019; De Brouwer et al., 2019; Deng et al., 2020; 2021) model state evolution and generate observation trajectories in continuous time, as shown in Fig. 1(c), but are unable to directly recover the sampling structure. In contrast, this paper directly learns from the original irregular sampling points, as shown in Fig. 1(d), which preserves the original sampling structure and avoids information loss caused by regularization or temporal alignment. However, preserving the original sampling-point structure still faces two challenges.

Challenge 1: It is challenging to capture real temporal dynamics under irregular sampling. Regular sequence and segment-aligned methods (Yoon et al., 2019; Jeon et al., 2022; Naiman et al., 2024b; Karami et al., 2025) use regularized representations that alter the original sampling process and introduce pseudo-observations, while continuous-time methods (Rubanova et al., 2019; De Brouwer et al., 2019; Deng et al., 2020; 2021) fail to characterize irregular sampling rhythms with a single smooth state evolution process. This challenge is further amplified in data-scarce scenarios, where limited samples make real temporal dynamics harder to learn.

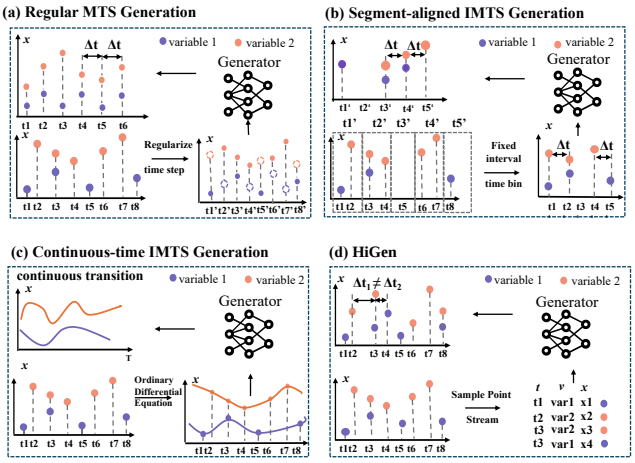


Figure 1: Comparison of different paradigms for irregular multivariate time-series generation.

Challenge 2: It is challenging to model variable interactions under asynchronous sampling while preserving the original sampling-point representation. Existing generation methods commonly train and generate different variables at unified timestamps or within predefined time bins (Yoon et al., 2019; Jeon et al., 2022; Naiman et al., 2024b; Karami et al., 2025). Regular-sequence and segment-aligned methods learn variable relationships from regularized representations, making the learned dependencies deviate from the real underlying correlations, while continuous-time models (Rubanova et al., 2019; De Brouwer et al., 2019; Deng et al., 2020; 2021) do not natively support variable interactions for generation tasks. Such limitations are further magnified in data-scarce scenarios, where inter-variable correlations are difficult to be adequately observed.

To address the above challenges, we propose HiGen, a hierarchical framework for irregular multivariate time series generation. **Addressing challenge 1:** For temporal dynamic modeling, HiGen learns the sampling-time structure of irregular sample points at two levels. At the window level, it learns the gap between consecutive non-empty windows to preserve sparse observation periods. At the point level, the model learns a set of ordered bin centers. During generation, the learned window level model determines when the next non-empty window appears, while the point level model selects a suitable time bin according to the current state and generates a concrete interval around that bin. This decomposition models temporal dynamics as a structured sampling process, which helps the model learn irregular timing patterns more stably when training samples are limited. **Addressing challenge 2:** HiGen models asynchronous variable relationships at the sample, window, and point levels. The sample level describes the overall state and general tendency, the window level uses historical windows to model cross-time dependencies, and the point level uses intra-window self-attention to capture local relationships among sample points. Thus, each sample point is generated by fusing sample, window, and point level information without forcing asynchronous observations into unified temporal positions. In low-data scenarios, this hierarchical relationship modeling shares contextual information across windows and sample points, using limited samples more effectively while reducing overfitting to common variables and local patterns.

We propose HiGen, a hierarchical framework for irregular multivariate time series generation. The main contributions are summarized as follows.

- We design a structured temporal modeling mechanism that captures coarse intervals between windows and local intervals through simultaneous decisions and learnable time bins.

- We introduce a multi-level conditioning mechanism to model asynchronous variable interactions across sample, window, and point level.
- Extensive experiments show that HiGen achieves better downstream utility and generation fidelity than state-of-the-art baselines.

2 Related Work

Regular time series generation. Adversarial models improve temporal consistency through recurrent architectures and supervised dynamics (Esteban et al., 2017; Yoon et al., 2019), with extensions covering causal optimal transport, attribute conditioning, frequency-domain modeling, contrastive imitation, Transformer-based generation, and variational frameworks (Xu et al., 2020; Lin et al., 2020; Alaa et al., 2021; Jarrett et al., 2021; Li et al., 2022; Desai et al., 2021). Diffusion-based and tokenized methods further use denoising, discrete temporal tokens, and image-domain representations (Chen et al., 2024; Naiman et al., 2024a). These methods mainly target regular sequences and often require preprocessing for irregular observations, which can alter sampling times and asynchronous variable relationships.

Irregular time series generation. Existing approaches model sampling time, missingness, continuous-time dynamics, and structured EHR records (Ramponi et al., 2018; Pei et al., 2021; Bing et al., 2022; Lu et al., 2024; Tian et al., 2024; Karami et al., 2025; 2024), while continuous-time models support arbitrary intervals (Chen et al., 2018; Rubanova et al., 2019; De Brouwer et al., 2019; Kidger et al., 2020; Li et al., 2020; Deng et al., 2020; 2021; Schirmer et al., 2022; Herrera et al., 2021). However, they often operate on processed time-feature matrices, masks, structured tokens, or continuous trajectories rather than original observation points, and do not necessarily model when observations occur, which variables are observed, and their values. HiGen instead directly generates time–variable–value points through a sample–window–point hierarchy, jointly modeling irregular sampling rhythms and asynchronous variable dependencies.

3 Methodology

3.1 Framework Overview

Problem formulation. Given a dataset $\mathcal{D} = \{\mathcal{S}_i\}_{i=1}^{I}$ of irregular multivariate time series, we represent each sample as an ordered sequence of sampling points:

$$\mathcal{S}_i = \langle s_{i,1}, \ldots, s_{i,L_i} \rangle, \qquad s_{i,j} = (t_{i,j}, v_{i,j}, x_{i,j}), \tag{1}$$

where L_i is the number of points, $t_{i,j}$ is the observation time, $v_{i,j} \in \mathcal{V}$ identifies the observed variable from the variable set $\mathcal{V}$, and $x_{i,j} \in \mathbb{R}$ is the observed value. Times satisfy $0 \le t_{i,1} \le \cdots \le t_{i,L_i} \le T_i$, where T_i is the observation horizon. Multiple variables may share a timestamp or be observed at different frequencies; an absent variable denotes an unobserved value, not a zero. We refer to sampling points as points below.

HiGen models each sample through the sample, window, and point levels shown in Fig. 2. At the sample level, the sample is encoded into a global representation c_i. At the window level, non-empty windows are connected by recurrent memory: before window k, the previous state $m^{+}_{i,k-1}$ evolves with the window gap $\Delta^{w}_{i,k}$ into $m^{-}_{i,k}$; after the window is encoded into summary $w_{i,k}$, the memory is updated to $m^{+}_{i,k}$. At the point level, conditioned on c_i and $m^{-}_{i,k}$, points inside window k are generated autoregressively, with learnable time bins.

During training, each real sample is encoded into c_i, which is also used to predict the sample label, and partitioned into non-empty windows. For each window, $m^{-}_{i,k}$ is obtained from $m^{+}_{i,k-1}$ and $\Delta^{w}_{i,k}$, the point decoder is trained on the real points, and $w_{i,k}$ updates the memory to $m^{+}_{i,k}$. During generation, the model samples c_i, predicts the synthetic label from c_i, initializes $m^{+}_{i,0}$, and repeatedly predicts the next non-empty window, evolves to $m^{-}_{i,k}$, generates its points autoregressively, summarizes them as $w_{i,k}$, and updates the memory to $m^{+}_{i,k}$ until termination.

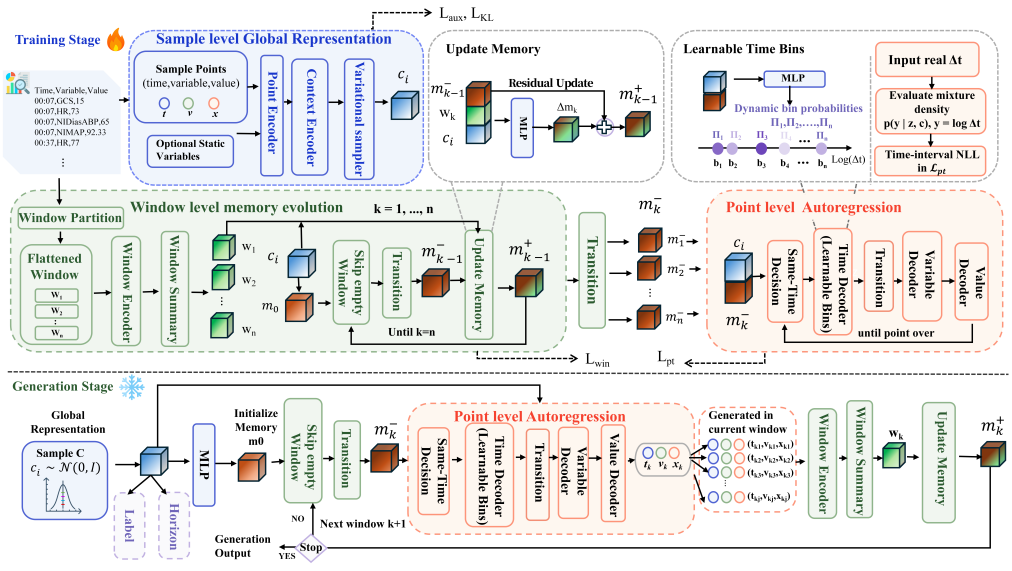


Figure 2: Overview of the proposed hierarchical irregular time-series generation framework.

3.2 Sample Level Global Representation

We design the sample level global representation module to provide a global condition for the whole irregular sample. Since a single window only contains partial observations, it cannot fully describe the overall observation horizon, variable occurrence tendency, and sample level temporal pattern. The sample is therefore encoded into a global representation c_i, which conditions both window level memory evolution and point level autoregression.

The point encoder f_{pt} maps each sample point $s_{i,j} = (t_{i,j}, v_{i,j}, x_{i,j})$ into a token representation by jointly embedding its time, variable identity, and observed value. The resulting point tokens are processed by a recurrent context encoder f_{ctx}, followed by masked pooling:

$$c_i^{\text{seq}} = f_{\text{ctx}}\left(f_{\text{pt}}(s_{i,1}), \ldots, f_{\text{pt}}(s_{i,L_i})\right). \tag{2}$$

If static variables are available, they are encoded into c_i^{stat} and added to the sequence summary, $\bar{c}_i = c_i^{\text{seq}} + c_i^{\text{stat}}$; otherwise, $\bar{c}_i = c_i^{\text{seq}}$.

To connect training and generation, we parameterize a variational posterior (Kingma & Welling, 2014) over the sample level global representation:

$$q_\phi(c_i \mid \mathcal{S}_i) = \mathcal{N}\left(\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2)\right), [\boldsymbol{\mu}_i, \log \boldsymbol{\sigma}_i^2] \quad = f_{\text{post}}(\bar{c}_i). \tag{3}$$

During training, c_i is sampled from the posterior via reparameterization and regularized toward $\mathcal{N}(0, I)$ with a KL term; during generation, it is sampled directly from this prior to reduce the train-generation mismatch.

The sample-level representation c_i initializes window memory, conditions both decoders, and, when supervision is available, predicts auxiliary targets such as the observation horizon or class label through $\mathcal{L}_{\text{aux}}$.

3.3 Window Level Memory Evolution

We design the window level memory evolution module to model coarse temporal evolution across separated observation periods. The sample is divided into an ordered sequence of non-empty windows, preserving the intervals between adjacent non-empty windows as window level time intervals.

Let $W_{i,k}$ denote the k-th non-empty window of sample i, with start time $T_{i,k}^w$. For a sample point inside this window, its local time is

$$\tau_{i,k,j} = t_{i,k,j} - T_{i,k}^w. \tag{4}$$

This separates absolute time from local window time: the window level process decides when a non-empty window appears in absolute time, while the point level process models relative timing and variable-value observations within the current window.

3.3.1 WINDOW ENCODER

The window encoder takes the point tokens inside $W_{i,k}$ as input and produces a window summary $w_{i,k}$ for the window level memory update. To preserve local structures, we use a structure-aware self-attention encoder with temporal, variable, and synchronization biases. For two sample points a and b in attention head h, the attention logit is

$$\alpha_{a,b}^{(h)} = \frac{(q_a^{(h)})^\top k_b^{(h)}}{\sqrt{d_h}} + b_{a,b}^{\mathrm{time},(h)} + b_{a,b}^{\mathrm{var},(h)} + b_{a,b}^{\mathrm{sync},(h)}. \tag{5}$$

Let τ_a and τ_b denote the local timestamps of sample points a and b. The temporal bias is

$$b_{a,b}^{\mathrm{time},(h)} = -\frac{|\tau_a - \tau_b|}{\tau_h}, \qquad \tau_h = \mathrm{softplus}(\eta_h), \tag{6}$$

where $\tau_h > 0$ controls how quickly the temporal bias decreases with the local time difference.

The variable bias captures compatibility between variable types:

$$b_{a,b}^{\mathrm{var},(h)} = (e_{v_a}^{(h)})^\top e_{v_b}^{(h)}, \tag{7}$$

where $e_{v_a}^{(h)}$ and $e_{v_b}^{(h)}$ are learnable variable-type embeddings for head h.

The synchronization bias highlights sample points observed at the same local timestamp:

$$b_{a,b}^{\mathrm{sync},(h)} = \gamma_h \mathbb{I}\left(|\tau_a - \tau_b| \le \epsilon_{\mathrm{sync}}\right), \tag{8}$$

where γ_h is a learnable head-specific coefficient and ϵ_{sync} is a small threshold. These biases allow the encoder to distinguish nearby, type-compatible, and synchronized sample points. The contextualized point tokens are then pooled into the window summary $w_{i,k}$.

3.3.2 WINDOW LEVEL MEMORY UPDATE

The window level memory connects non-empty windows and carries historical information from previous windows. Let $m_{i,k-1}^{+}$ be the memory after the previous non-empty window. For $k > 1$, the window level gap is

$$\Delta_{i,k}^{w} = T_{i,k}^{w} - T_{i,k-1}^{w}, \tag{9}$$

with $\Delta_{i,1}^{w}$ measured from the sample start time. The memory is initialized and updated as

$$m_{i,0}^{+} = f_{\mathrm{init}}(c_i), \quad m_{i,k}^{-} = f_{\mathrm{trans}}(m_{i,k-1}^{+}, \Delta_{i,k}^{w}, c_i), \quad m_{i,k}^{+} = f_{\mathrm{upd}}(m_{i,k}^{-}, w_{i,k}, c_i). \tag{10}$$

Here, $m_{i,k}^{-}$ denotes the pre-window memory before window k, while $m_{i,k}^{+}$ is the post-window memory after incorporating $w_{i,k}$ and is propagated to the next non-empty window. The transition function moves the memory forward according to elapsed time, while the update function incorporates the current window summary. The update is parameterized in a residual form, and during generation $m_{i,k}^{-}$ also conditions the next window gap and local termination.

3.4 POINT LEVEL AUTOREGRESSION

The point level autoregression models fine-grained sample-point dependencies inside each non-empty window. Given the pre-window memory $m_{i,k}^{-}$, the local decoding state $z_{i,k,j-1}$ is initialized from $m_{i,k}^{-}$ and updated with the teacher-forced prefix. Let $q_{j-1} = [z_{i,k,j-1}; c_i]$. The next-point distribution is

$$p(s_j \mid \cdot) = p(r_j \mid q_{j-1})\, p(\delta_j \mid r_j, q_{j-1})\, p(v_j \mid q_{j-1})\, p(x_j \mid v_j, q_{j-1}). \tag{11}$$

Here, $r_{i,k,j}$ indicates whether the current point shares the previous local timestamp. If $r_{i,k,j} = 1$, $\delta_{i,k,j} = 0$; otherwise, the interval is modeled by the time decoder. The remaining factors predict the variable type and its conditional value, allowing both synchronized observations and asynchronous local timing patterns.

3.4.1 LEARNABLE TIME BIN DECODER

For $r_{i,k,j} = 0$, we model $y_{i,k,j} = \log \delta_{i,k,j}$ using K ordered learnable bin centers:

$$b_1 = b_0, \qquad b_r = b_{r-1} + \mathrm{softplus}(g_r), \quad r = 2, \ldots, K. \tag{12}$$

Here, b_0 and $\{g_r\}_{r=2}^{K}$ are learnable parameters. The decoder predicts mixture weights

$$\boldsymbol{\pi}_{i,k,j} = \mathrm{softmax}\left(f_\pi(z_{i,k,j-1}, c_i) + \log \boldsymbol{\rho}\right), \tag{13}$$

where $\boldsymbol{\rho} \in (0,1)^K$ denotes learnable bin gates. The conditional log-interval density is

$$p(y_{i,k,j} \mid \cdot) = \sum_{r=1}^{K} \pi_{i,k,j,r} \mathcal{N}\left(y_{i,k,j}; b_r + \mu_{i,k,j,r}, \sigma^2_{i,k,j,r}\right), \delta_{i,k,j} \quad = \exp(y_{i,k,j}). \tag{14}$$

The predicted residual mean and standard deviation adapt each temporal prototype to the current state, allowing interval patterns at different temporal scales.

3.5 TRAINING OBJECTIVE AND FLATTENED-WINDOW LEARNING

Training uses teacher forcing on real sample points. All non-empty windows in a batch are flattened for window encoding and teacher-forced point losses, while the window level memory retains the original window order of each sample. The point level loss is

$$\mathcal{L}_{\mathrm{pt}} = -\sum_{i,k,j} \mathbb{I}^{\mathrm{valid}}_{i,k,j} \log p_\theta\left(s_{i,k,j} \mid m^{-}_{i,k}, c_i, s_{i,k,<j}\right), \tag{15}$$

where $s_{i,k,<j}$ is the teacher-forced prefix and $\mathbb{I}^{\mathrm{valid}}_{i,k,j}$ masks padded positions. $\mathcal{L}_{\mathrm{win}}$ supervises the next non-empty window and current window termination, $\mathcal{L}_{\mathrm{KL}}$ regularizes the sample level representation toward the standard Gaussian prior, and $\mathcal{L}_{\mathrm{aux}}$ contains optional sample level label or horizon supervision. The final objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{pt}} + \lambda_{\mathrm{win}}\mathcal{L}_{\mathrm{win}} + \lambda_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}} + \lambda_{\mathrm{aux}}\mathcal{L}_{\mathrm{aux}}. \tag{16}$$

3.6 GENERATION PROCEDURE

After training, $c_i \sim \mathcal{N}(0, I)$ initializes the window-level memory. For each window, its temporal position and current memory are determined first. Conditioned on the memory and c_i, point-level autoregression generates each point by deciding whether it shares the previous local timestamp; otherwise, a local interval is sampled from the learnable time-bin decoder, followed by the variable and its conditional value. The generated window is encoded into $w_{i,k}$ to update memory, while absolute timestamps are recovered from the window start time and local times. Generation stops at the predicted horizon.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and baselines. We conduct experiments on three representative irregular time-series datasets, with classification as the downstream task, including P12 (Silva et al., 2012; Luo et al., 2018), Activity (Vidulin et al., 2010), and CesNet (Koumar et al., 2025). P12 contains clinical physiological records. Activity contains wearable sensor localization recordings collected during daily activities. CesNet contains aggregated network flow-level statistics. These datasets cover medical monitoring, human activity sensing, and network traffic analysis. We compare HiGen with TimEHR (Karami et al., 2025), TimeDiff (Tian et al., 2024), GT-GAN (Jeon et al., 2022), and KoVAE (Naiman et al., 2024b), covering image-based, diffusion, adversarial, and variational generation. Each baseline uses its native preprocessing.

Implementation and metrics. We report the average results over three stratified runs with different random seeds. Following existing generation methods, downstream classification utility is measured by AUROC and AUPRC (macro-averaged for multiclass data) (Davis & Goadrich, 2006).

Generation fidelity is measured by precision, recall, density, and coverage (PRDC) (Naeem et al., 2020; Karami et al., 2025); temporal irregularity is measured by the Jensen–Shannon divergence between normalized log-interval distributions (NLI-JSD, lower is better) (Lin, 1991). Synthetic labels are generated by the corresponding model, while test-set labels are used only for final downstream evaluation. All methods receive the same label supervision without test-label leakage. Further implementation and label-generation details are provided in Appendix C.

4.2 Downstream Utility under Data Scarcity

The full real training set contains N_{train} samples, and the real test set contains N_{test} samples. For each ratio $r \in \{5\%, 10\%, 15\%\}$, we obtain $\mathcal{D}_r$ by class-stratified sampling, with $|\mathcal{D}_r| \approx rN_{\text{train}}$. All methods use the same $\mathcal{D}_r$ within each run, while the real testing set N_{test} is used only for downstream evaluation. To assess whether generated data preserve discriminative information and can complement limited real observations, we evaluate synthetic-only training and synthetic augmentation as described below.

Synthetic-only training. The generator is trained on $\mathcal{D}_r$ and produces N_{train} synthetic samples for training an LSTM classifier, which is tested on N_{test}. Table 1 shows that HiGen achieves the best AUROC and AUPRC in every r setting. The gains are substantial on P12 and Activity and remain consistent on CesNet, indicating that the generated sequences preserve discriminative information even with scarce training data. A comparison against classifiers trained on real subsets is reported in Appendix D, Table 5.

Table 1: Downstream classification performance under different training ratios.

Dataset	Ratio	Metric	Ours	TimEHR	TimeDiff	GT-GAN	KoVAE
P12	5%	AUROC	**0.6770±.0296**	0.5497±.0285	0.4745±.0171	0.4848±.0076	0.5519±.0378
		AUPRC	**0.2420±.0150**	0.1467±.0093	0.1245±.0072	0.2227±.0291	0.1887±.0302
	10%	AUROC	**0.7186±.0161**	0.6859±.0218	0.4729±.0440	0.4672±.0079	0.6600±.0424
		AUPRC	**0.2350±.0128**	0.2156±.0246	0.1273±.0148	0.1336±.0103	0.2194±.0332
	15%	AUROC	**0.7725±.0056**	0.6916±.0105	0.5275±.0686	0.5645±.0238	0.6666±.0278
		AUPRC	**0.3054±.0117**	0.2515±.0178	0.1585±.0366	0.2029±.0240	0.2360±.0357
CesNet	5%	AUROC	**0.9851±.0077**	—	0.5680±.0609	0.7035±.0346	0.7758±.0098
		AUPRC	**0.9575±.0157**	—	0.3181±.0536	0.4688±.0516	0.5264±.0178
	10%	AUROC	**0.9946±.0037**	0.6803±.0120	0.5548±.0531	0.7723±.0388	0.8747±.0253
		AUPRC	**0.9834±.0091**	0.4709±.0263	0.2798±.0531	0.5125±.0796	0.6826±.0401
	15%	AUROC	**0.9980±.0012**	0.7738±.0163	0.5321±.0489	0.8311±.0186	0.7354±.0404
		AUPRC	**0.9926±.0043**	0.5277±.0141	0.2746±.0396	0.6092±.0676	0.4714±.0344
Activity	5%	AUROC	**0.8070±.0420**	0.6231±.0085	0.5027±.0106	0.6241±.1657	0.5949±.0488
		AUPRC	**0.6804±.0429**	0.5190±.0149	0.3718±.0035	0.4937±.1690	0.5093±.0483
	10%	AUROC	**0.9011±.0199**	0.6200±.0377	0.5894±.0248	0.7693±.0658	0.7449±.0611
		AUPRC	**0.8072±.0405**	0.5271±.0323	0.4372±.0121	0.6134±.0563	0.6243±.0503
	15%	AUROC	**0.9243±.0569**	0.7148±.0324	0.4926±.0193	0.6533±.0528	0.7764±.0645
		AUPRC	**0.8294±.0754**	0.5581±.0181	0.3848±.0371	0.5015±.0450	0.6363±.0624

Note: "—" indicates unable to generate valid synthetic samples due to input data scarcity.

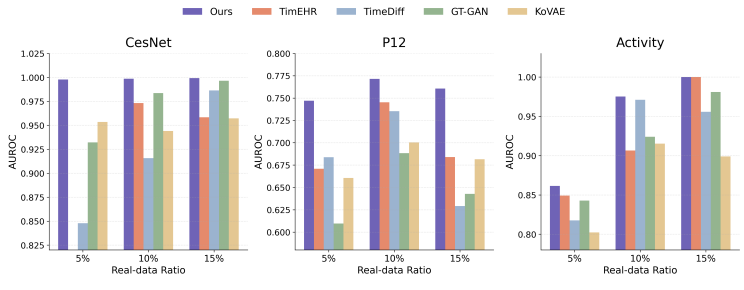


Figure 3: AUROC under synthetic augmentation.

Synthetic augmentation. We generate $N_{\text{train}} - |\mathcal{D}_r|$ synthetic samples. Together with $\mathcal{D}_r$, each training set still contains N_{train} instances. As shown in Fig. 3, HiGen reports the best AUROC at all ratios on P12, CesNet and Activity. TimEHR at 5% is unavailable because it did not produce enough valid samples. Overall, the generated data effectively complement limited real observations.

4.3 Generation Fidelity and Temporal Irregularity Analysis

Following prior time series generation studies, we evaluate generation fidelity by comparing real and synthetic distributions using PRDC (Karami et al., 2025), aiming to assess the preservation of both value dynamics and irregular sampling patterns. We train a label-free LSTM autoencoder on real training sequences and use its final hidden state to compare real and synthetic samples. Precision and density measure sample realism, while recall and coverage measure distributional coverage and diversity; PR-F1 summarizes their balance. NLI-JSD further compares normalized log-interval distributions between adjacent sampling points to evaluate irregular sampling rhythms. Table 2 shows that HiGen achieves the best overall generation fidelity and the lowest NLI-JSD across all three datasets.

Table 2: Generation fidelity comparison.

Dataset	Model	PR-F1 ↑	Prec. ↑	Rec. ↑	Dens. ↑	Cov. ↑	NLI-JSD ↓
P12	TimeDiff	0.4933	0.4516	0.5434	0.3565	0.2015	0.7565
	TimEHR	0.5239	0.5918	0.4700	0.5296	0.3957	0.6992
	KoVAE	0.0261	0.4482	0.0135	0.3366	0.0225	0.8527
	GT-GAN	0.0085	0.4459	0.0043	0.3326	0.0218	0.8527
	Ours	**0.6409**	**0.6430**	**0.6388**	**0.5594**	**0.6364**	**0.1608**
Activity	TimeDiff	0.2973	0.1872	**0.7230**	0.1111	0.1303	0.7696
	TimEHR	0.2992	0.2744	0.3289	0.2124	0.2027	0.8550
	KoVAE	0.0485	0.2385	0.0270	0.1530	0.0507	0.8549
	GT-GAN	0.2121	0.2184	0.2061	0.1357	0.0506	0.8549
	Ours	**0.6588**	**0.6703**	0.6477	**0.5673**	**0.6514**	**0.2889**
CesNet	TimeDiff	0.4006	0.3146	0.5515	0.2810	0.4496	0.4637
	TimEHR	0.4590	0.5401	0.3990	0.4749	0.4603	0.6850
	KoVAE	0.1069	0.4472	0.0607	0.4189	0.2075	0.8550
	GT-GAN	0.2755	0.3823	0.2154	0.3453	0.2024	0.8550
	Ours	**0.6327**	**0.6333**	**0.6321**	**0.5464**	**0.6284**	**0.0162**

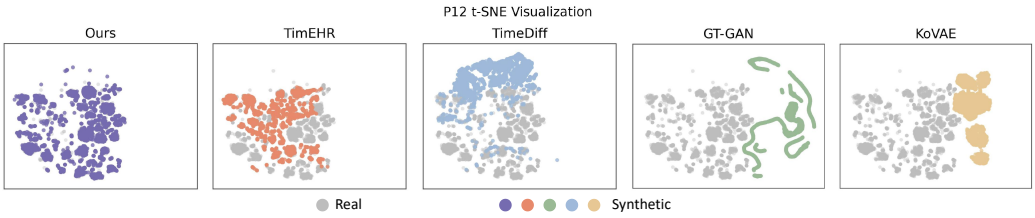


Figure 4: t-SNE visualization of real and generated P12 samples.

4.4 Qualitative Visualization

We compute a joint t-SNE projection (van der Maaten & Hinton, 2008) over real samples and all generated samples. Figure 4 shows that HiGen overlaps with and covers more of the real P12 manifold, whereas several baselines exhibit distribution shifts or collapsed regions.

4.5 Variable Interaction Recovery

To further evaluate whether the generated data preserve interactions among variables, we compare same-time variable co-occurrence matrices on P12 using 5% of the training data. Here, "same-time" means that two variables are observed at exactly the same timestamp within a sample. For each sample, let T_i and T_j denote the timestamps of variables i and j. Their co-occurrence is measured by the Jaccard similarity $|T_i \cap T_j|/|T_i \cup T_j|$, where a larger value means that the two variables are more often observed together. The matrix error is the average absolute difference between the generated and real matrices for comparable pairs of different variables. The Real panel is the reference, so its error is 0.00%. As shown in Fig. 5, HiGen obtains an error of 4.00%, compared with 19.26% for TimeEHR, reducing the error by 79.2%. This indicates that HiGen more accurately preserves the

shared observation patterns of variables under irregular sampling. Further analysis of asynchronous cross-variable dependencies at different temporal lags is provided in Appendix F.

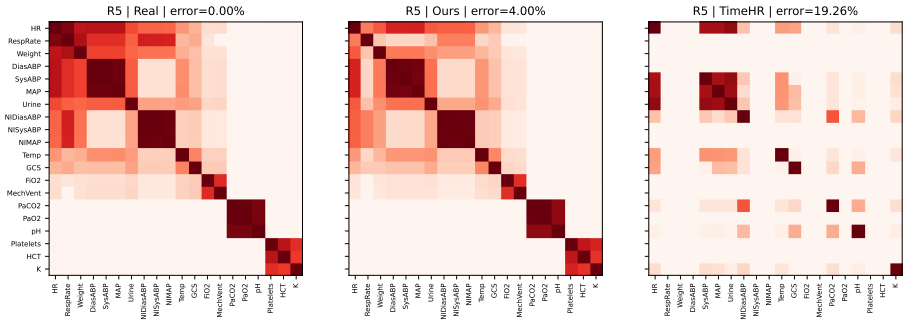


Figure 5: Variable interaction recovery on P12 under the 5% training-data setting.

4.6 Ablation Study

All variants follow the synthetic-only training setting using the full real training set and generate N_{train} synthetic samples. Table 3 shows that the full model obtains the best or tied-best AUROC, AUPRC, and PR-F1 on every dataset. Removing window memory reduces all three metrics on every dataset. Removing attention bias yields the same PR-F1 as the full model on P12 but lower AUROC and AUPRC. On Activity and CesNet, removing any component reduces PR-F1. Overall, the results support the effectiveness of the full model across both downstream utility and generation fidelity.

Table 3: Ablation study on key components.

Dataset	Variant	AUROC	AUPRC	PR-F1
P12	w/o Time bins	0.7225	0.2691	0.6121
	w/o Attn. bias	0.7639	0.3753	0.6409
	w/o Var. sample rep.	0.6511	0.2029	0.6274
	w/o Window memory	0.7205	0.2284	0.6356
	Ours	**0.7816**	**0.4188**	**0.6409**
Activity	w/o Time bins	0.9774	0.9664	0.6407
	w/o Attn. bias	0.9977	0.9885	0.6553
	w/o Var. sample rep.	0.9595	0.9039	0.6154
	w/o Window memory	0.9570	0.8636	0.6096
	Ours	**0.9977**	**0.9900**	**0.6588**
CesNet	w/o Time bins	0.9986	0.9958	0.6320
	w/o Attn. bias	0.9992	0.9971	0.6281
	w/o Var. sample rep.	0.9992	0.9971	0.6313
	w/o Window memory	0.9994	0.9977	0.6208
	Ours	**0.9996**	**0.9985**	**0.6327**

4.7 Parameter Sensitivity

We vary one hyperparameter at a time on P12 under the full-data synthetic-only protocol. Figure 6 shows that HiGen remains generally stable across the tested settings, with only moderate fluctuations in AUROC, precision, and recall. AUPRC is relatively more sensitive to hyperparameter changes, but remains consistently strong around the best settings. Overall, the model is robust to reasonable hyperparameter variations.

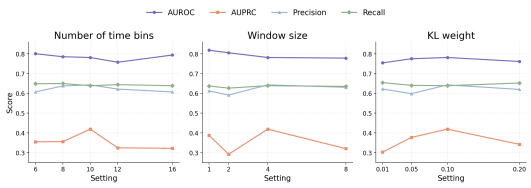


Figure 6: Parameter sensitivity analysis on the P12 dataset.

5 Conclusion

We propose HiGen, a hierarchical framework for irregular multivariate time series generation. HiGen models temporal dynamics through inter-window gaps and learnable local time bins, and captures asynchronous variable interactions across the three levels.

AI USE STATEMENT

We acknowledge the use of ChatGPT to polish the language, improve readability, and refine the visual presentation of figures in this paper. The use of this tool was limited to grammatical and stylistic refinement as well as polishing framework figures; the authors have verified the accuracy of the content and accept full responsibility for the final submission.

ETHICS STATEMENT

This work studies irregular multivariate time series generation and does not involve the recruitment of human subjects or the collection of new personal data. Experiments use existing benchmark datasets, including P12, Activity, and CesNet. We do not release new datasets containing personally identifiable information.

REPRODUCIBILITY STATEMENT

The HiGen architecture and training procedure are described in Section 3, and the experimental settings and evaluation protocols are provided in Section 4. Dataset descriptions, implementation details, and additional results are provided in the appendix. Results are reported over three stratified runs with different random seeds. Further implementation details are provided in Appendix C. Code repository: [https://anonymous.4open.science/r/HiIMTS-GEN-release-02FF].

REFERENCES

Ahmed Alaa, Alex Chan, and Mihaela van der Schaar. Generative time-series modeling with fourier flows. In *International Conference on Learning Representations*, 2021.

Simon Bing, Andrea Dittadi, Stefan Bauer, and Patrick Schwab. Conditional generation of medical time series for extrapolation to underrepresented populations. *PLOS Digital Health*, 1(7): e0000074, 2022. doi: 10.1371/journal.pdig.0000074.

Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. BRITS: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, 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):6085, 2018.

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

Zhicheng Chen, Shibo Feng, Zhong Zhang, Xi Xiao, Xingyu Gao, and Peilin Zhao. SDformer: Similarity-driven discrete transformer for time series generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 132179–132207, 2024. doi: 10.52202/079017-4201.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *International Conference on Machine Learning*, pp. 233–240, 2006.

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

R. Deng, B. Chang, M. Brubaker, G. Mori, and A. Lehrmann. Modeling continuous stochastic processes with dynamic normalizing flows. In *Advances in Neural Information Processing Systems*, 2020.

Ruizhi Deng, Marcus A. Brubaker, Greg Mori, and Andreas M. Lehrmann. Continuous latent process flows. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5162–5173, 2021.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. TimeVAE: A variational auto-encoder for multivariate time series generation, 2021. arXiv:2111.08095.

Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued medical time series generation with recurrent conditional GANs, 2017. arXiv:1706.02633.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Data augmentation using synthetic data for time series classification with deep residual networks, 2018. arXiv:1808.02455.

Tal Gonen, Itai Pemper, Ilan Naiman, Nimrod Berman, and Omri Azencot. Time series generation under data scarcity: A unified generative modeling approach. In *Advances in Neural Information Processing Systems*, volume 38, pp. 191446–191485, 2025. doi: 10.52202/085713-5750.

Calypso Herrera, Florian Krach, and Josef Teichmann. Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering. In *International Conference on Learning Representations*, 2021.

Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Time-series generation by contrastive imitation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 28968–28982, 2021.

Jinsung Jeon, Jeonghak Kim, Haryong Song, Seunghyeon Cho, and Noseong Park. GT-GAN: General purpose time series synthesis with generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2022.

Hojjat Karami, David Atienza, and Anisoara Ionescu. SynEHRgy: Synthesizing mixed-type structured electronic health records using decoder-only transformers, 2024. arXiv:2411.13428.

Hojjat Karami, Mary-Anne Hartley, David Atienza, and Anisoara Ionescu. TimEHR: Image-based time series generation for electronic health records. *IEEE Journal of Biomedical and Health Informatics*, 29(12):9170–9180, 2025. doi: 10.1109/JBHI.2025.3577328.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Josef Koumar, Karel Hynek, Tomáš Čejka, and Pavel Šiška. CESNET-TimeSeries24: Time series dataset for network traffic anomaly detection and forecasting. *Scientific Data*, 12(1):338, 2025. doi: 10.1038/s41597-025-04603-x.

Xiaomin Li, Vangelis Metsis, Huangyingrui Wang, and Anne Hee Hiong Ngu. TTS-GAN: A transformer-based time-series generative adversarial network. In *Artificial Intelligence in Medicine*, pp. 133–143. Springer, 2022. doi: 10.1007/978-3-031-09342-5_13.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using GANs for sharing networked time series data: Challenges, initial promise, and open questions. In *ACM Internet Measurement Conference*, 2020.

Chang Lu, Chandan K. Reddy, Ping Wang, Dong Nie, and Yue Ning. Multi-label clinical time-series generation via conditional GAN. *IEEE Transactions on Knowledge and Data Engineering*, 36(4): 1728–1740, 2024. doi: 10.1109/TKDE.2023.3310909.

Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, and Xiaojie Yuan. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 31, pp. 1603–1614, 2018.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pp. 7176–7185, 2020.

Ilan Naiman, Nimrod Berman, Itai Pemper, Idan Arbiv, Gal Fadlon, and Omri Azencot. Utilizing image transforms and diffusion models for generative modeling of short and long time series. In *Advances in Neural Information Processing Systems*, volume 37, pp. 121699–121730, 2024a.

Ilan Naiman, N. Benjamin Erichson, Pu Ren, Michael W. Mahoney, and Omri Azencot. Generative modeling of regular and irregular time series data via koopman VAEs. In *International Conference on Learning Representations*, 2024b.

Hengzhi Pei, Kan Ren, Yuqing Yang, Chang Liu, Tao Qin, and Dongsheng Li. Towards generating real-world time series data. In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 469–478, 2021. doi: 10.1109/ICDM51629.2021.00058.

Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. T-CGAN: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling, 2018. arXiv:1811.08295.

Yulia Rubanova, Tian Qi Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, volume 32, pp. 5321–5331, 2019.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19388–19405. PMLR, 2022.

Satya Narayan Shukla and Benjamin M. Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2019.

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*, pp. 245–248, 2012.

Muhang Tian, Bernie Chen, Allan Guo, Shiyi Jiang, and Anru R. Zhang. Reliable generation of privacy-preserving synthetic electronic health record time series via diffusion models. *Journal of the American Medical Informatics Association*, 31(11):2529–2539, 2024. doi: 10.1093/jamia/ocae229.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Ashish Vaswani et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Vedrana Vidulin, Mitja Lustrek, Bostjan Kaluza, Rok Piltaver, and Jana Krivec. Localization data for person activity. UCI Machine Learning Repository, 2010.

Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. In *International Joint Conference on Artificial Intelligence*, 2021.

Tianlin Xu, Li K. Wenliang, Michael Munn, and Beatrice Acciaio. COT-GAN: Generating sequential data via causal optimal transport. In *Advances in Neural Information Processing Systems*, 2020.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2019.

A Training and Generation Procedures

Algorithms 1 and 2 summarize the flattened-window training and hierarchical generation procedures, respectively. Let M_i denote the number of non-empty windows in sample i, and let Δ_w denote the base-window width.

Algorithm 1 Training with Flattened Window Batches

Input: Real streams $\{\mathcal{S}_i\}_{i=1}^{B}$, base-window width Δ_w
Output: Training loss $\mathcal{L}$

Infer posterior conditions $q_\phi(c_i \mid \mathcal{S}_i)$ and sample c_i
Partition each stream into ordered non-empty windows
Flatten windows and pack teacher-forced event tensors
Encode windows in parallel to obtain summaries $w_{i,k}$
Encode causal event prefixes to obtain local contexts
Initialize $m^+_{i,0} \leftarrow f_{\text{init}}(c_i)$
for $k = 1, \ldots, \max_i M_i$ **do**
  Select streams containing the k-th window
  Derive the ground-truth skip count $s^*_{i,k}$
  Compute the real window gap $\Delta^w_{i,k}$
  $m^-_{i,k} \leftarrow f_{\text{trans}}(m^+_{i,k-1}, \Delta^w_{i,k}, c_i)$
  Store $m^-_{i,k}$ for teacher-forced decoding
  $m^+_{i,k} \leftarrow f_{\text{upd}}(m^-_{i,k}, w_{i,k}, c_i)$
end for
for each flattened window (i, k) in parallel **do**
  Initialize local state $z_{i,k,0} \leftarrow m^-_{i,k}$
  for each valid event position j **do**
    Construct the teacher-forced next-point targets
    Accumulate the point-level loss $\mathcal{L}_{\text{pt}}$
    Update $z_{i,k,j}$ using the observed event
  end for
  Accumulate the window-level loss $\mathcal{L}_{\text{win}}$
end for
Compute the KL loss $\mathcal{L}_{\text{KL}}$
Compute the auxiliary loss $\mathcal{L}_{\text{aux}}$
Normalize losses over valid events and windows
$\mathcal{L} \leftarrow \mathcal{L}_{\text{pt}} + \lambda_{\text{win}}\mathcal{L}_{\text{win}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}}$
return $\mathcal{L}$

Algorithm 2 Hierarchical Generation Procedure

Input: Trained parameters θ
Output: Synthetic stream $\hat{\mathcal{S}}$

Sample $c \sim \mathcal{N}(0, I)$ and predict horizon $\hat{T}$
Initialize $m^+_0 \leftarrow f_{\text{init}}(c)$
$\hat{\mathcal{S}} \leftarrow \emptyset$, $k \leftarrow 1$
while the predicted horizon $\hat{T}$ is not reached **do**
  Determine the next non-empty window position T^w_k
  Obtain pre-window memory m^-_k from m^+_{k-1}, the window gap, and c
  Initialize local state $z_{k,0} \leftarrow m^-_k$, $\hat{W}_k \leftarrow \emptyset$, and $j \leftarrow 1$
  while the current window is not terminated **do**
    Sample same-time decision $\hat{r}_{k,j}$
    if $\hat{r}_{k,j} = 1$ **then**
      $\hat{\delta}_{k,j} \leftarrow 0$
    else
      Sample $\hat{y}_{k,j}$ from the learnable time-bin decoder
      $\hat{\delta}_{k,j} \leftarrow \exp(\hat{y}_{k,j})$
    end if
    Update local time using $\hat{\delta}_{k,j}$
    Recover absolute timestamp $\hat{t}_{k,j}$ from T^w_k and the local time
    Sample variable $\hat{v}_{k,j}$ and conditional value $\hat{x}_{k,j}$
    Append $(\hat{t}_{k,j}, \hat{v}_{k,j}, \hat{x}_{k,j})$ to $\hat{W}_k$ and $\hat{\mathcal{S}}$
    Update local state $z_{k,j}$
    $j \leftarrow j + 1$
  end while
  Encode $\hat{W}_k$ into window summary $\hat{w}_k$
  $m^+_k \leftarrow f_{\text{upd}}(m^-_k, \hat{w}_k, c)$
  $k \leftarrow k + 1$
end while
return $\hat{\mathcal{S}}$

B Description of Datasets

P12. P12 (Silva et al., 2012; Luo et al., 2018) is derived from the PhysioNet/Computing in Cardiology Challenge 2012 and contains multivariate clinical measurements collected from patients admitted to intensive care units. Each patient record consists of irregularly observed physiological variables during the first 48 hours after admission, where different variables are measured at different frequencies and timestamps. Following the preprocessing used in prior irregular time-series studies, we retain 37 temporal variables and obtain 3,997 patient sequences. The resulting dataset contains 3,597 training samples and 400 test samples and is used for binary classification. The average number of observed sampling points per sequence is 435.14, with an average observation horizon of 47.26 hours.

Table 4: Dataset statistics.

Dataset	#Samples	#Classes	#Vars.	Avg. #Points	Avg. Horizon	Max Horizon
P12	3997	2	37	435.14	47.26	48.00
Activity	376	3	12	913.04	10028.62	35976.11
CesNet	420	5	10	121.62	45.39	47.98

Activity. Activity (Vidulin et al., 2010) is derived from the UCI Localization Data for Person Activity dataset. It contains three-dimensional positional measurements collected from wearable sensor tags attached to different body locations. The resulting multivariate time series contain 12 variables

and are naturally irregular because measurements from different sensors are not always observed at exactly aligned timestamps. After preprocessing, we obtain 376 sequences, including 301 training samples and 75 test samples, covering three activity classes. Activity contains dense sequences, with an average of 913.04 observed sampling points per sample and an average observation horizon of 10,028.62 in the original time unit.

CesNet. CesNet (Koumar et al., 2025) is a network-traffic dataset containing aggregated flow level statistics collected from multiple network entities. We construct each sample as an irregular multivariate sequence of network measurements and use 10 traffic-related variables. Class labels are derived from the corresponding network-segment names, resulting in a five-class classification task. After preprocessing, the dataset contains 420 sequences, including 340 training samples and 80 test samples. Each sequence contains 121.62 sampling points on average, with an average observation horizon of 45.39. This dataset provides a non-clinical irregular time-series setting in which sampling patterns and variable interactions reflect network traffic behavior.

C Implementation Details

HiGen is implemented in PyTorch and trained with AdamW on an NVIDIA RTX 3090 GPU with 24GB memory. Model dimensions are selected according to dataset scale and sequence length, and KL warm-up is used for the variational sample-level representation. The generator uses $d_{\text{model}} = 128$, $z_{\text{dim}} = 128$, $c_{\text{dim}} = 128$, and $h_{\text{dim}} = 256$, with a time-encoding dimension of 32, an outcome-embedding dimension of 32, and two progress features. AdamW uses a weight decay of 10^{-6} and gradient clipping at 1.0. For P12, we use $K = 10$ time bins, learning rate 10^{-3}, batch size 32, and a KL warm-up of 11,000 steps. For Activity, we use $K = 8$, learning rate 5×10^{-4}, batch size 16, and a KL warm-up of 11,000 steps. For CesNet, we use $K = 10$, learning rate 5×10^{-4}, batch size 32, and a KL warm-up of 8,000 steps. The window widths are expressed in the corresponding dataset time units. We use the shared evaluation protocol described in Section 4.1, including stratified sampling at each data ratio, a fixed test set, and averaging over three random seeds.

All competing methods are trained on exactly the same ratio-specific real training subsets as HiGen. We retain each baseline's original input representation and do not introduce artificial missingness, timestamp jitter, interpolation, or any other post-processing that could improve fidelity externally. No test samples are used during generator training or output conversion. All methods generate the same target number of samples and are evaluated using the same downstream classifier and fidelity pipeline.

All compared generators use labels only from the corresponding real training subset. Real labels are never copied to synthetic samples; instead, each method produces synthetic labels through its native generation mechanism. HiGen predicts the outcome from the sampled sample-level representation c_i, whereas GT-GAN, KoVAE, and TimeDiff generate the outcome jointly with the sequence. TimEHR generates the outcome as part of its static conditional representation or samples it from the training-label distribution when static covariates are unavailable. During downstream evaluation, the classifier target is obtained from each method's generated output and is excluded from the classifier input. Test labels are used only for final evaluation.

To ensure a reproducible comparison, we report the main dataset-specific configurations used for the baseline models.

For KoVAE, we used the model with hidden dimension 20 and latent dimension 16. The learning rate was 7×10^{-4}, the KL weight was 0.1, the prior-prediction weight was 5×10^{-4}, and the reconstruction weight was 1.0. The model was trained for 600 epochs with sequence length 128 and stride 64.

For TimeDiff, we used its configuration with hidden dimension 256, three recurrent layers, embedding dimension 64, and time-embedding dimension 256. The learning rate was 8×10^{-5}, the number of diffusion steps was 1000, and dropout was disabled.

For GT-GAN, the hidden dimension was 24. The batch sizes for the 5%, 10%, 15%, and 100% settings were 16, 24, 32, and 32, respectively. All baseline models were trained and sampled using the same dataset-specific training subsets and generated-sample protocol as the proposed method.

For TimEHR, we used the public TimEHR implementation, which represents each irregular time series as a two-channel image containing the value matrix and the observation mask. TimEHR consists of two conditional GAN modules. The first module is a conditional CWGAN-GP that generates the observation mask, while the second module is a conditional Pix2Pix model that generates values conditioned on the generated mask. For the CWGAN-GP module, we used 150 training epochs, one critic update per generator update, learning rate 3×10^{-4}, gradient penalty weight 10, latent dimension 100, kernel size 4, generator feature width 16, critic feature width 16, and Adam optimizers with $\beta_1 = 0$ and $\beta_2 = 0.9$. For the Pix2Pix module, we used 150 training epochs, one discriminator update per generator update, learning rate 2×10^{-3}, kernel size 4, generator feature width 64, discriminator feature widths $(8, 16)$, value reconstruction weight 200. The generator and discriminator used Adam with $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

D Real-Only versus Synthetic-Only Training

Table 5 compares classifiers trained on matched real subsets with those trained on synthetic samples generated from the same subsets. Synthetic training is comparable or better in most low-data settings, particularly on CesNet, although real data remain stronger in several P12 and Activity settings.

Table 5: Real-only and synthetic-only downstream performance.

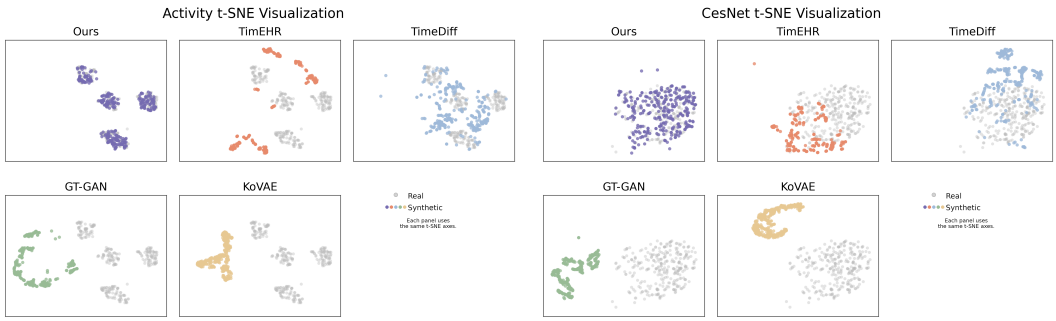
Dataset	Ratio	Real AUROC	Real AUPRC	Synthetic AUROC	Synthetic AUPRC
P12	100%	0.7402	0.3100	0.7816	0.4188
	5%	0.6948	0.2603	0.6770	0.2420
	10%	0.6815	0.2280	0.7186	0.2350
	15%	0.6578	0.2303	0.7725	0.3053
CesNet	100%	0.9997	0.9988	0.9996	0.9985
	5%	0.8604	0.6328	0.9851	0.9575
	10%	0.9176	0.7700	0.9946	0.9834
	15%	0.9566	0.8762	0.9980	0.9926
Activity	100%	1.0000	1.0000	0.9977	0.9900
	5%	0.7745	0.6637	0.8070	0.6804
	10%	0.8534	0.7372	0.9011	0.8072
	15%	0.9803	0.9373	0.9243	0.8294

Let N_{train} denote the full training-set size and $n_r = |\mathcal{D}_r|$ the number of available real samples at ratio r. The real-only classifier is trained on these n_r samples. For synthetic-only training, the generator learns from the same real subset and produces N_{train} synthetic samples for classifier training. Both classifiers are evaluated on the same held-out real test set. This comparison measures the utility of expanding a limited real dataset through generation; it is not an equal-size comparison between real and synthetic training sets.

Sampling from the learned distribution may expose the classifier to varied realizations of patterns supported by the available records and reduce overfitting. These potential benefits do not imply that synthetic samples provide additional independent real-world observations, and they are not observed in every dataset–ratio setting.

E Qualitative Distribution Analysis via t-SNE

We further provide a qualitative comparison between the real and synthetic data distributions using t-SNE. Specifically, we encode each real or synthetic irregular time series into a sample level representation using the same label-free LSTM encoder as in the fidelity evaluation, and then project these representations into a shared two-dimensional space. Gray points denote real samples and colored points denote synthetic samples. The real-sample coordinates are fixed across methods, and all panels share the same t-SNE axes, making relative overlap and distribution shift directly comparable.



(a) Activity **(b) CesNet**

Figure 7: t-SNE visualization of real and synthetic sequence embeddings on (a) Activity and (b) CesNet. Gray points represent real samples, while colored points represent synthetic samples generated by different methods. For each dataset, all method panels share the same t-SNE coordinate system and fixed real-sample coordinates.

As shown in Fig. 7(a), HiGen closely aligns with the major real-data clusters on Activity and preserves their multimodal structure. TimEHR generates several clusters outside the real-data support, while GT-GAN and KoVAE exhibit substantial distribution shifts. TimeDiff covers a broader region, but many synthetic samples are dispersed between or outside the real clusters.

A similar pattern is observed on CesNet (Fig. 7(b)). HiGen provides substantial overlap with both the dense central region and less populated parts of the real distribution. TimEHR captures only part of the real support, while TimeDiff shows noticeable shifts toward regions outside the main real-data manifold. GT-GAN and KoVAE exhibit clearer separation between real and synthetic samples, indicating more limited distributional coverage.

Overall, the visualizations show that HiGen achieves both close distributional alignment and broad coverage of real-data modes, rather than matching only a small subset of the real distribution. These observations are consistent with the quantitative PRDC results, where HiGen provides a favorable balance between fidelity and coverage.

F Asynchronous Variable Interaction Recovery

The main paper evaluates same-time variable co-occurrence. Here we further examine whether HiGen preserves asynchronous cross-variable dependencies across different temporal lags on P12, Activity, and CesNet under the 5% training-data setting. For each dataset, the corresponding 5% real training subset is used as the reference.

For each sample, let $\tau_1 < \tau_2 < \cdots < \tau_M$ denote the ordered non-empty observation timestamps, and let $\mathcal{V}_n$ denote the set of variables observed at τ_n. For two adjacent timestamps, we define the positive temporal lag as $\Delta\tau_n = \tau_{n+1} - \tau_n > 0$. For each dataset separately, we divide the positive lags into four bins using the 25th, 50th, and 75th percentiles of the real-data lag distribution:

$$B_1 = (0, q_{25}], \qquad B_2 = (q_{25}, q_{50}], \qquad B_3 = (q_{50}, q_{75}], \qquad B_4 = (q_{75}, \infty).$$

Thus, B_1–B_4 represent progressively longer intervals between adjacent observation timestamps.

For each lag bin B_ℓ, we construct a directed variable-interaction matrix $M^{(\ell)}$. Its entry (i, j) measures the probability that variable j is observed at the next timestamp given that variable i is observed at the current timestamp and $\Delta\tau_n \in B_\ell$:

$$M^{(\ell)}_{i,j} = \frac{\sum_n \mathbb{I}[i \in \mathcal{V}_n,\, j \in \mathcal{V}_{n+1},\, \Delta\tau_n \in B_\ell]}{\sum_n \mathbb{I}[i \in \mathcal{V}_n,\, \Delta\tau_n \in B_\ell]}. \tag{17}$$

When multiple variables occur at either timestamp, all corresponding source–target pairs are counted. Diagonal entries ($i = j$) are excluded so that the analysis focuses on interactions between different variables.

We compare each generated matrix with the corresponding real matrix using mean absolute error (MAE):

$$E_\ell = \frac{1}{|\Omega_\ell|} \sum_{(i,j)\in\Omega_\ell} \left| M^{(\ell)}_{\text{real},i,j} - M^{(\ell)}_{\text{syn},i,j} \right|, \tag{18}$$

where Ω_ℓ denotes the comparable off-diagonal entries. Lower values indicate better recovery of lagged variable interactions. For each method, we also report an overall MAE by averaging E_ℓ over the four lag bins.

Figure 8 summarizes the lag-specific interaction errors on P12, Activity, and CesNet. Each panel corresponds to one dataset, and B_1–B_4 represent increasingly larger adjacent observation intervals. Lower MAE indicates that the generated cross-variable interaction matrix is closer to the corresponding real matrix. HiGen maintains substantially lower errors across the lag bins on all three datasets, showing more faithful recovery of asynchronous cross-variable dependencies.

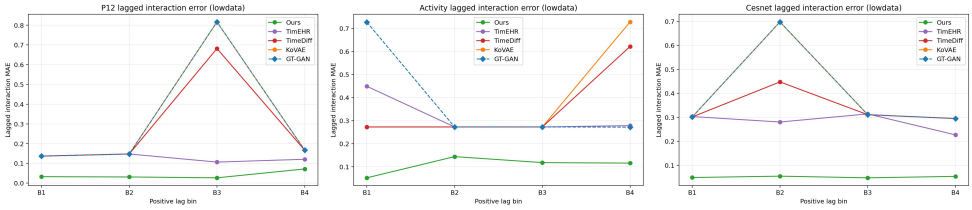


Figure 8: Lagged variable-interaction error on P12, Activity, and CesNet under the 5% training-data setting. Each panel reports MAE across four positive lag bins B_1–B_4, from shorter to longer temporal intervals. Lower is better.

Interaction heatmaps. Figures 9–11 visualize the interaction matrices underlying the MAE results. For each dataset, the four heatmap rows correspond to B_1–B_4. Within each matrix, the vertical axis denotes the variable observed at the current timestamp (source variable), and the horizontal axis denotes the variable observed at the next timestamp (target variable). Each cell represents the conditional probability that the target variable is observed after the source variable within the corresponding lag range; darker cells indicate stronger interactions. The columns compare the real reference with HiGen and the corresponding baseline. A generated matrix closer in structure and intensity to the real matrix indicates better recovery of lagged variable dependencies.

P12. As shown in Fig. 9, HiGen preserves the major interaction structures of the real data across all four lag ranges, including the dominant variable-specific patterns. In contrast, TimEHR produces much sparser interactions, particularly in several lag bins. This structural difference is consistent with the lower MAE of HiGen in Fig. 8.

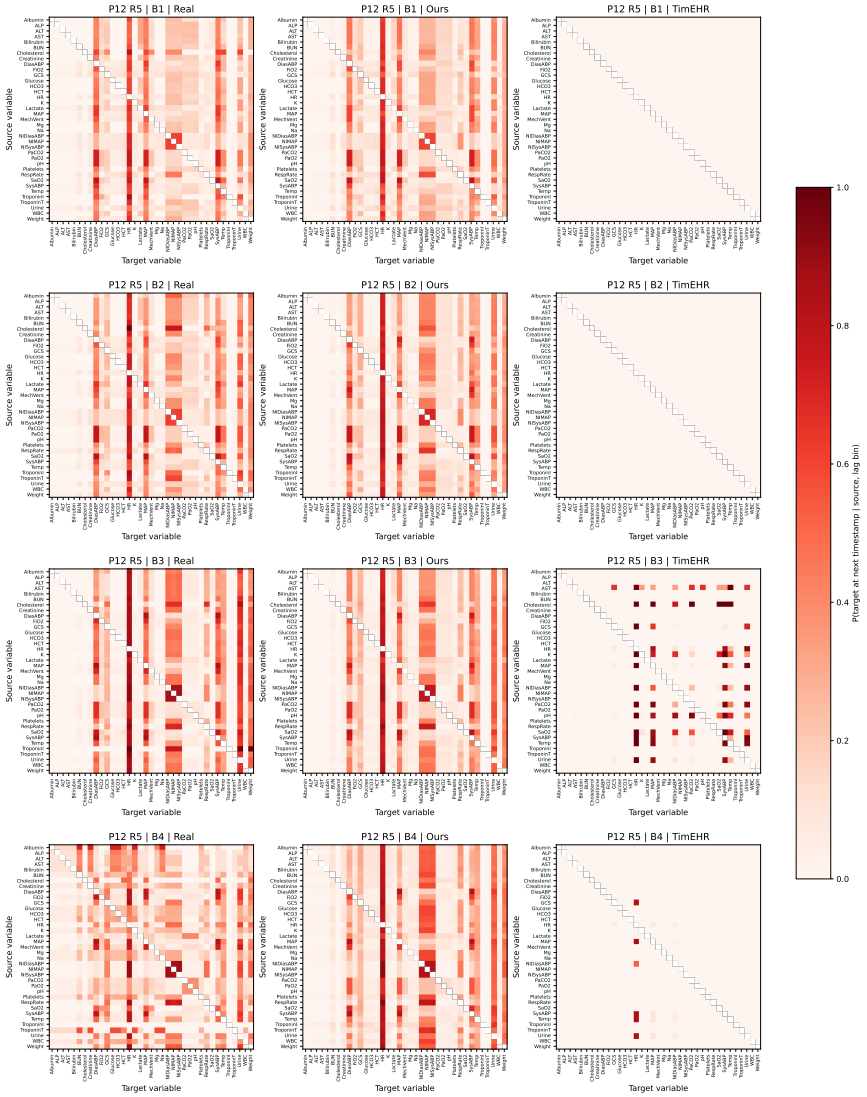


Figure 9: Lagged variable-interaction matrices on P12 under the 5% training-data setting. Rows correspond to lag bins B_1–B_4, while columns compare the real reference, HiGen, and TimEHR.

Activity. Figure 10 shows the corresponding interaction patterns on Activity. Although its adjacent intervals are concentrated around a dominant sampling interval, the four bins still capture different ranges of this interval distribution. HiGen reproduces substantially more of the real cross-variable structure, whereas the baseline exhibits larger deviations in several lag bins.

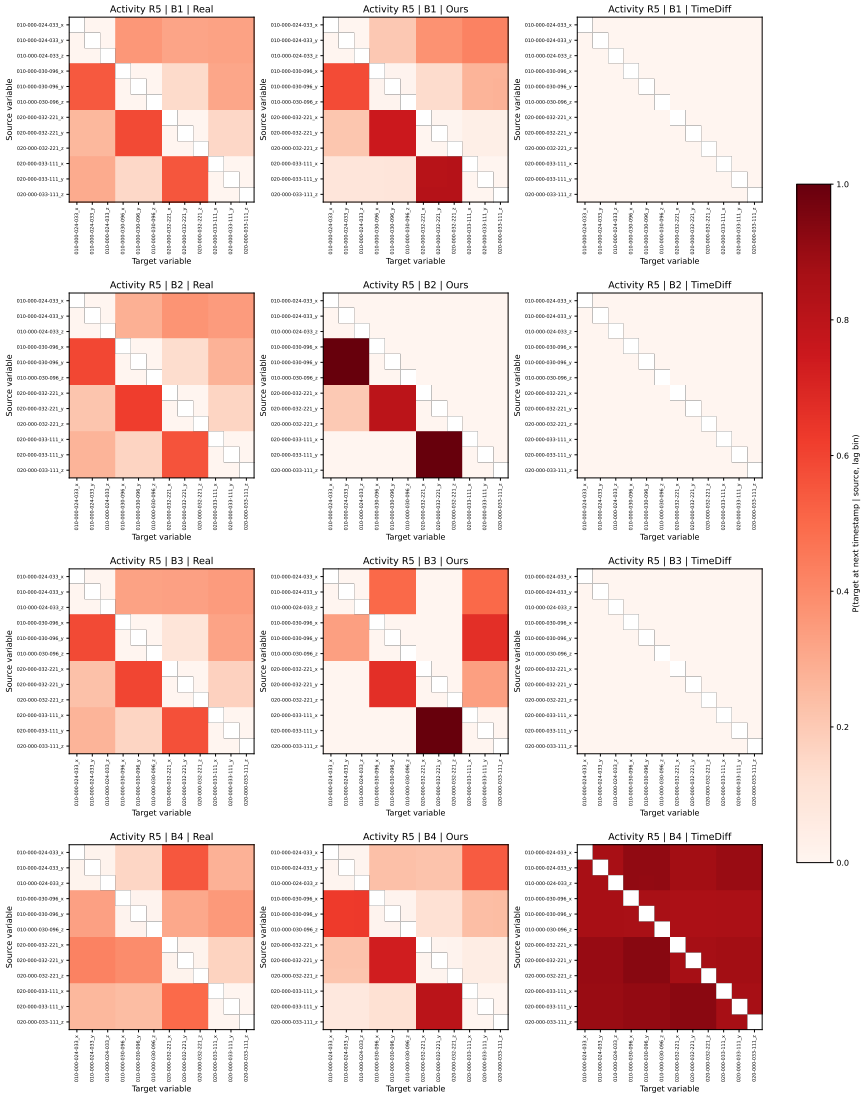


Figure 10: Lagged variable-interaction matrices on Activity under the 5% training-data setting. Rows correspond to lag bins B_1–B_4.

CesNet. As shown in Fig. 11, the real CesNet data exhibit dense but non-uniform transitions among variables. HiGen preserves these patterns across the four lag ranges, while the baseline concentrates interactions within fewer temporal ranges and leaves other bins substantially underrepresented. This explains the consistently lower lagged interaction error of HiGen.

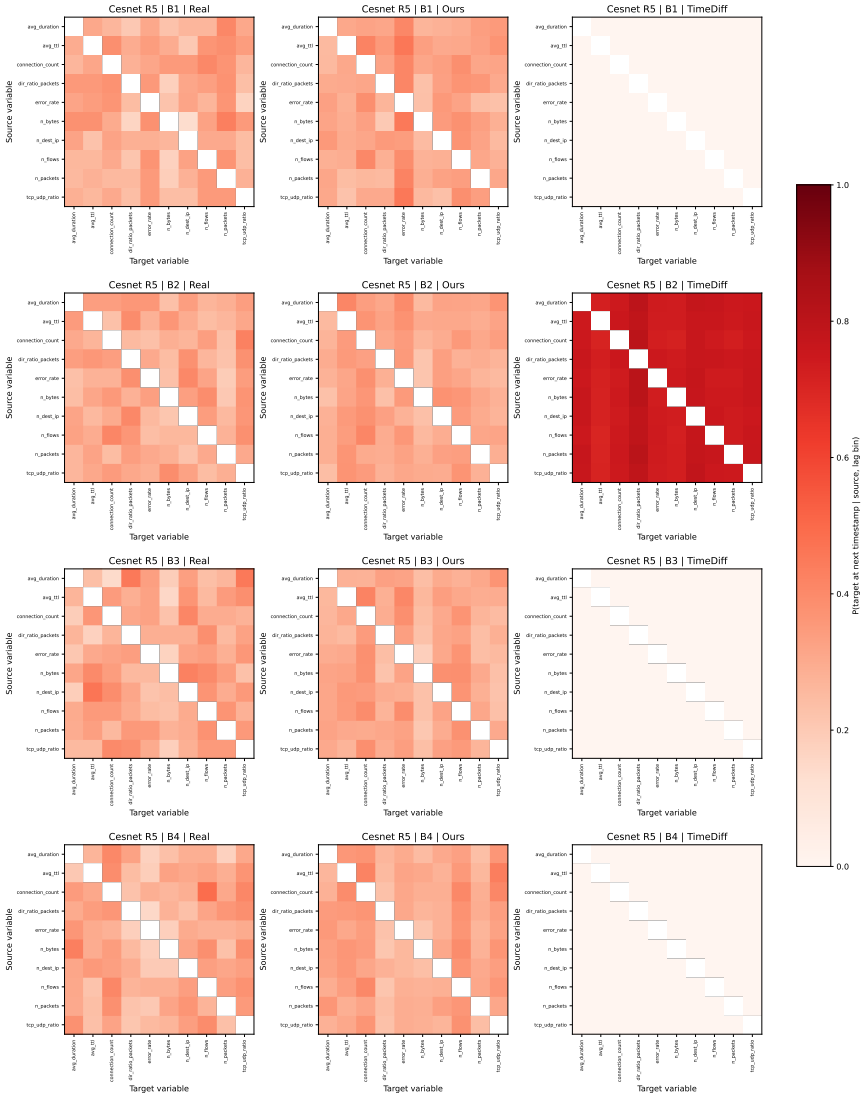


Figure 11: Lagged variable-interaction matrices on CesNet under the 5% training-data setting. Rows correspond to lag bins B_1–B_4.

Overall, the MAE and heatmap analyses consistently show that HiGen better preserves which variables tend to follow others across different observation intervals. Together with the same-time co-occurrence analysis in the main paper, these results evaluate both same-time observation patterns and asynchronous cross-time variable interactions.

G Parameter Sensitivity

Activity. Figure 12 shows that HiGen is generally stable across different numbers of time bins, window sizes, and KL weights on Activity. AUROC and AUPRC remain close to 1.0 throughout the tested ranges, while precision and recall exhibit only moderate variations. The main fluctuation occurs in recall for larger window sizes, whereas downstream utility remains consistently high. Overall, HiGen is robust to reasonable hyperparameter changes on Activity.

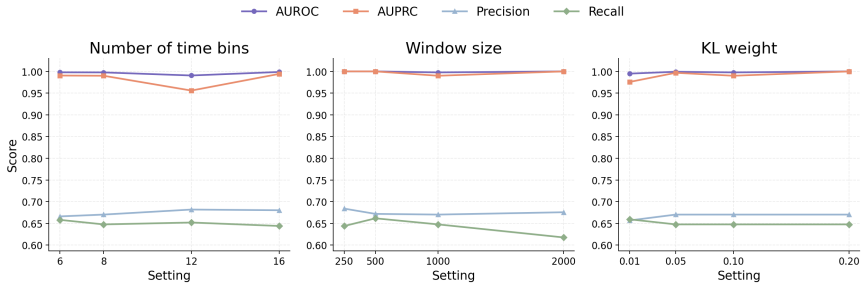


Figure 12: Parameter sensitivity analysis on the Activity dataset.

CesNet. Figure 13 shows even stronger stability on CesNet. AUROC and AUPRC remain nearly saturated across all tested settings, while precision and recall vary only slightly. Changing the number of time bins, window size, or KL weight therefore has limited influence on overall performance, indicating low sensitivity to these hyperparameters on CesNet.

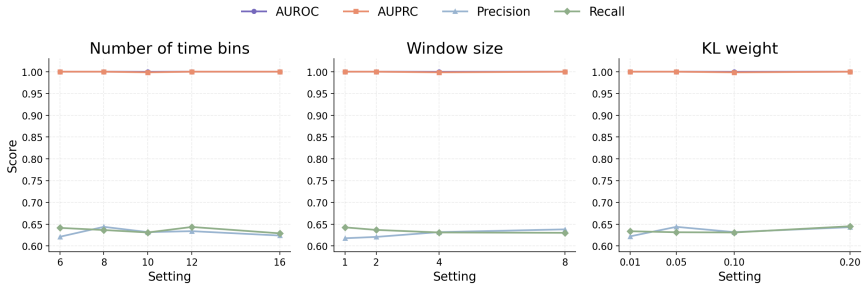


Figure 13: Parameter sensitivity analysis on the CesNet dataset.

H LIMITATIONS

HiGen currently focuses on unconditional generation of irregular multivariate time series and does not explicitly support user-specified controllable generation, such as generating samples according to desired attributes or natural-language instructions. Extending the framework with explicit control signals or instruction-based conditioning is an interesting direction for future work.