CT-GSSN: A Continuous-Time Graph State-Space Network for Irregularly Sampled Multivariate Time Series

Arnav Jain
Unaffiliated
arnav0422@gmail.com

Abstract

Modeling real-world dynamic systems is profoundly challenging due to the dual complexities of irregular sampling and multivariate inter-dependencies. A vast amount of critical time series data, from healthcare to climate science, is collected at non-uniform intervals with misaligned observations across variables. Concurrently, the evolution of any single series is often intricately linked to the state of others. Existing deep learning models typically address only one of these challenges, creating a bifurcation in research between models for irregular data and those for relational systems. This paper introduces the Continuous-Time Graph State-Space Network (CT-GSSN), the first framework to synergistically unify the relational inductive biases of Graph Neural Networks, the continuous-time dynamics of Neural Ordinary Differential Equations, and the linear-time efficiency of modern State-Space Models, all under a provable Lyapunov stability guarantee[1]*. This hybrid design allows it to model complex inter-series dynamics on a graph while handling arbitrary time gaps with linear complexity. We introduce a Lyapunov-based stability regularizer into a multi-task self-supervised pre-training strategy to learn generalizable and robust representations for forecasting, imputation, and classification. Experiments on challenging benchmarks including MIMIC-III, PhysioNet, and METR-LA show that CT-GSSN achieves state-of-the-art performance, improving forecasting accuracy by up to 12% over strong baselines like GraphSSM and Neural CDEs, establishing a new standard for robust and scalable modeling of irregular temporal graphs.*

[1]**Clarification:** Our guarantees are precise: (i) a *Lyapunov-style stability along encountered trajectories* when the symmetric part of the Jacobian is non-positive (encouraged by $\mathcal{L}_{\text{stable}}$), and (ii) *per-interval global contraction* under the constrained parameterization in Sec. 10. We add a uniform switching result in Sec. 17.

1. Introduction

The paradigm of pre-trained models, large-scale networks trained on diverse data, has achieved transformative success in natural language and vision [1]. This success is inspiring a new frontier in time series analysis, with models aiming to provide general-purpose representations for tasks like forecasting and anomaly detection [1–3]. However, the majority of these models are architected for an idealized world of clean, complete, and regularly sampled data [2]. This overlooks two fundamental and co-occurring challenges that define the nature of most real-world time series.

The first is **irregular sampling**. Data from critical domains such as healthcare, climate science, and IoT is rarely collected at uniform intervals [4–6]. Observations are often asynchronous across different variables, with inconsistent time gaps and missing values [4, 7, 8]. Standard deep learning architectures like RNNs and Transformers, which are inherently discrete, struggle to process data where time itself is a continuous, variable dimension [4, 9, 10].

The second is **multivariate inter-dependency**. Real-world phenomena are generated by complex systems of interacting components [11]. In a multivariate time series, the evolution of one series is often dependent on others [3, 12]. Graph Neural Networks (GNNs) have emerged as the premier tool for this challenge, modeling time series as nodes in a graph and their interactions as edges [12–16].

These two challenges are not independent; they are facets of a single, deeper problem: modeling complex dynamic systems in their native form. Current research is bifurcated: one community develops models for irregular data (e.g., based on Neural ODEs [9, 18, 49]), while another develops GNNs for regularly sampled relational data [12, 15, 19]. This separation is a critical bottleneck. The quadratic complexity of Transformers makes them ill-suited for the long, sparse sequences of irregular data [2, 6], and GNNs that rely on these backbones inherit their limitations. The central, unsolved problem is thus the development of a *single, unified general-purpose model* that is natively continuous-time, relational, and computationally efficient for large-scale pre-training.

This paper introduces the **Continuous-Time Graph State-Space Network (CT-GSSN)**, a novel architecture that synthesizes innovations from three fields. It leverages (1) the relational power of GNNs to capture inter-series dependencies [3, 15, 20], (2) the linear-time complexity of modern State-Space Models (SSMs) like Mamba [5, 6, 12, 19] for scalability, and (3) a continuous-time formulation inspired by Neural ODEs to handle arbitrary sampling intervals [9, 21, 22]. While some works have combined two of these elements [9, 21, 23], none have achieved a true synthesis of all three with provable stability guarantees. CT-GSSN fills this gap, offering a principled solution to a long-standing challenge in time series analysis by making each component necessary and complementary: SSMs provide efficiency for long sequences where Transformers fail; GNNs capture relational structure that independent SSMs ignore; and the continuous-time ODE formulation handles irregular sampling that discrete models cannot.

Our primary contributions are:

1. **A Novel Architecture:** The proposal of the CT-GSSN architecture, a new model class that synergistically integrates Graph Neural Networks (GNNs), continuous-time Ordinary Differential Equations (ODEs), and State-Space Models (SSMs) for learning on graph-structured time series.
2. **A Formal Theoretical Guarantee:** The derivation of a Lyapunov-based stability regularizer and a formal theorem that provides, for the first time, a provable guarantee of stability for the learned continuous-time dynamics of a graph neural network.
3. **A Versatile Multi-Task Framework:** A tailored multi-task self-supervised pre-training strategy, including the novel Lyapunov-based regularizer, that promotes the learning of stable and generalizable dynamics for forecasting, imputation, and classification.
4. **State-of-the-Art Empirical Performance:** Extensive empirical validation on a diverse suite of benchmarks from healthcare, traffic analysis, and climate science, demonstrating that CT-GSSN achieves state-of-the-art performance against a wide range of strong baselines, including GraphSSM and Neural CDEs.

2. Related Work

Our work is positioned at the intersection of three active research areas: modeling irregularly sampled time series, graph-based forecasting, and efficient sequence modeling.

Models for Irregular Time Series. A significant body of work has focused on adapting deep learning models for irregular data. Early methods like GRU-D [7] introduced decay mechanisms to handle missingness. A more principled approach emerged with models based on Neural Ordinary Differential Equations (ODEs), which treat the hidden state as a continuous trajectory [9, 18, 49]. Latent-ODE [4] and its variants use an RNN to encode observations into a latent space, which is then evolved forward in time by an ODE solver. GRU-ODE-Bayes [26] offers a robust alternative, particularly in healthcare. More recent works like mTAN [3] and ContiFormer [14] have adapted the attention mechanism to a continuous-time setting, while Neural Controlled Differential Equations (Neural CDEs) [27] represent the state-of-the-art by leveraging the mathematical theory of controlled differential equations. While powerful, these models typically treat variables as independent and do not explicitly model the relational structure between them.

Graph Neural Networks for Time Series. Spatio-Temporal GNNs (STGNNs) have become the standard for multivariate forecasting on relational data, with pioneering applications in traffic forecasting [13, 19]. Models like GraphSTAGE [3, 15] and RAINDROP [6] use GNNs to capture spatial dependencies, often in conjunction with a temporal backbone like a TCN or RNN. A significant advancement in this area is the development of Graph Neural ODEs, which combine GNNs with continuous-time dynamics to model systems where both the graph structure and node states evolve over time [18, 22, 28–30]. However, the vast majority of STGNNs still presuppose regular, synchronous sampling across all nodes, making them unsuitable for the irregular data common in domains like healthcare [6]. TGNN4I [5, 21] is a notable exception that combines GNNs with an ODE-based GRU, but it does not leverage the efficiency of modern SSMs or provide formal stability guarantees.

State-Space Models for Time Series. Recently, State-Space Models (SSMs), particularly the Mamba architecture [5, 10, 19], have emerged as a highly efficient and effective alternative to Transformers for sequence modeling [2, 6]. Mamba's selective SSM mechanism achieves linear-time complexity while capturing long-range dependencies [6, 19, 51]. This has led to adaptations for time series, such as MambaTS [11] and TSMamba [12]. These models, however, are designed for regularly sampled data and do not inherently handle irregular timestamps or model explicit graph structures. GraphSSM [31] is a pioneering work that applies SSMs to temporal graphs but operates in a discrete-time framework, lacking a native continuous-time formulation. Unlike GraphSSM, which uses discrete updates on graph snapshots, our model handles true temporal irregularity via continuous ODE integration. GrassNet [22] combines GNNs and SSMs, but for the task of designing spectral graph filters, not for continuous-time forecasting. CT-GSSN is the first to integrate all three paradigms—GNNs, selective SSMs, and a continuous-time ODE formulation—into a unified, provably stable architecture for irregularly sampled, multivariate time series.

3. Background and Preliminaries

Our work builds upon concepts from three distinct but complementary fields: Graph Neural Networks for relational modeling, Neural Ordinary Differential Equations for continuous-time dynamics, and State-Space Models for efficient sequence processing.

Definition 3.1 (Irregular Multivariate Time Series (IMTS))**.** *An IMTS is a set of observations for N variables (or channels), $S = \{S_1, \dots, S_N\}$. Each variable S_n is a sequence of observation pairs $S_n = \{(t_{n,i}, x_{n,i})\}_{i=1}^{L_n}$, where $t_{n,i} \in \mathbb{R}^+$ is the timestamp and $x_{n,i} \in \mathbb{R}^{d_x}$ is the observed value. The timestamps are not uniform, and for any given time t, the set of observed variables may be a subset of the total variables [20, 50]. The relational structure between variables can be represented by a dynamic graph $\mathcal{G}_t = (V, E_t)$, where V is the set of variables and E_t is the set of edges at time t.*

Definition 3.2 (Graph Neural Network Layer)**.** *A GNN layer updates a node v's feature vector $\mathbf{h}_v$ via an aggregation of its neighbors $\mathcal{N}_v$:*

$$\mathbf{h}_v^{(l+1)} = \phi^{(l)}\left(\mathbf{h}_v^{(l)}, \bigoplus_{u\in\mathcal{N}_v} \psi^{(l)}(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)})\right)$$

where ϕ and ψ are differentiable functions (e.g., MLPs) and $\bigoplus$ is a permutation-invariant aggregation function (e.g., sum, mean) [14, 47, 48]. GNNs provide a powerful inductive bias for data with relational structure.

Definition 3.3 (Neural Ordinary Differential Equation (Neural ODE))**.** *A Neural ODE models the continuous evolution of a hidden state $\mathbf{h}(t)$ by parameterizing its derivative with a neural network f_θ:*

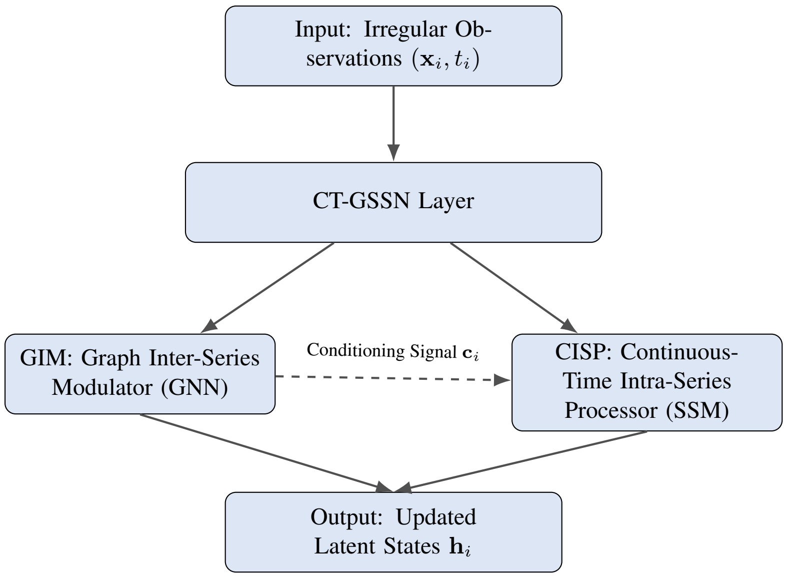
$$\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), t)$$

Given an initial state $\mathbf{h}(t_0)$, the state at any later time t_1 is found by solving this initial value problem: $\mathbf{h}(t_1) = \mathbf{h}(t_0) + \int_{t_0}^{t_1} f_\theta(\mathbf{h}(t), t)dt$. This provides a natural framework for handling data with irregular time gaps, as the model is defined over a continuous time domain [4, 49].

Definition 3.4 (Linear State-Space Model (SSM))**.** *A continuous-time linear SSM is defined by the ordinary differential equations (ODEs) [2]:*

$$\dot{\mathbf{h}}(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{h}(t) + \mathbf{D}\mathbf{u}(t)$$

where $\mathbf{h}(t) \in \mathbb{R}^D$ is the latent state, $\mathbf{u}(t) \in \mathbb{R}^P$ is the input, $\mathbf{y}(t) \in \mathbb{R}^Q$ is the output, and $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are state matrices [22]. Modern SSMs like Mamba [10, 19] use selective mechanisms to make these matrices input-dependent, achieving linear-time complexity for long sequence modeling.



Input: Irregular Observations $(\mathbf{x}_i, t_i)$
CT-GSSN Layer
GIM: Graph Inter-Series Modulator (GNN)
Conditioning Signal $\mathbf{c}_i$
CISP: Continuous-Time Intra-Series Processor (SSM)
Output: Updated Latent States $\mathbf{h}_i$

Figure 1. High-level view of a single CT-GSSN layer. The GIM processes inter-series relationships to generate a conditioning signal that modulates the dynamics of the CISP, which processes intra-series temporal evolution.

4. The CT-GSSN Architecture

The CT-GSSN is a novel deep learning architecture designed as a general-purpose model for irregularly sampled multivariate time series. It is formulated as a Graph Neural Ordinary Differential Equation where the vector field is parameterized by a selective, graph-modulated state-space operator. It consists of a stack of CT-GSSN layers, each unifying the modeling of inter-series (spatial) and intra-series (temporal) dynamics in a continuous-time framework.

4.1. Graph-based Inter-Series Modulator (GIM)

The GIM models the relational structure between time series. It is a GNN that operates on the graph of nodes, capturing spatial or cross-channel dependencies. At each step, the GIM can dynamically learn an adjacency matrix, allowing it to capture time-varying correlations. For each node i, the GIM computes a conditioning vector $\mathbf{c}_i$ by aggregating information from its neighborhood $\mathcal{N}_i$:

$$\mathbf{c}_i = \text{AGG}\left(\{\mathbf{m}_{j\to i} \mid j \in \mathcal{N}_i\}\right)$$

where the message $\mathbf{m}_{j\to i} = \text{MLP}(\mathbf{h}_i, \mathbf{h}_j)$ is a function of the hidden states of the source and target nodes. This vector $\mathbf{c}_i$ is not a direct prediction but a control signal that modulates the temporal dynamics of node i.

4.2. Continuous-Time Intra-Series Processor (CISP)

The CISP models the temporal dynamics within each series, conditioned by the GIM. The evolution of the hidden state for each node, $\mathbf{h}_i(t)$, is governed by a Neural ODE. The input to the system, $\mathbf{u}_i(t)$, represents the observed value for node i at time t (treated as an impulse):

$$\frac{d\mathbf{h}_i(t)}{dt} = f_\Theta(\mathbf{h}_i(t), \mathbf{c}_i(t), \mathbf{u}_i(t))$$

The core innovation is that the vector field f_Θ is parameterized by a structured State-Space Model (SSM). This provides a principled solution to irregular sampling, as the state at any future time can be found by integrating this ODE. Given an observation at time t_k after a previous one at t_{k-1}, the state evolution over the interval $\Delta t = t_k - t_{k-1}$ can be solved analytically for a linear system, avoiding costly numerical solvers:

$$\mathbf{h}(t_k) = e^{\mathbf{A}\Delta t}\mathbf{h}(t_{k-1}) + (\mathbf{A}^{-1}(e^{\mathbf{A}\Delta t} - \mathbf{I}))\mathbf{B}\mathbf{u}_k$$

This provides a principled solution to irregular sampling, superior to methods that wrap a discrete model in an external ODE solver.

4.3. Hybridization and Dynamics Modulation

The fusion of the GIM and CISP occurs through a novel hybridization mechanism. The conditioning vector $\mathbf{c}_i$ from the GIM is used to dynamically parameterize the state matrices $\mathbf{A}$ and $\mathbf{B}$ of the CISP for node i. This is achieved via small MLPs that project the conditioning vector to the appropriate matrix dimensions, producing dense matrices for maximum flexibility:

$$\mathbf{A}_i(t) = \text{MLP}_A(\mathbf{c}_i(t))$$
$$\mathbf{B}_i(t) = \text{MLP}_B(\mathbf{c}_i(t))$$

This design creates a deeply integrated architecture where the relational structure (GNN) directly governs the continuous-time temporal dynamics (SSM) of each series. This is a more fundamental approach to spatiotemporal modeling than simply stacking separate spatial and temporal blocks [3]. The full dynamic equation for a node i becomes:

$$\frac{d\mathbf{h}_i(t)}{dt} = \mathbf{A}_i(t)\mathbf{h}_i(t) + \mathbf{B}_i(t)\mathbf{u}_i(t)$$

This formulation can also be viewed through the lens of optimal control theory [32], where the GIM provides the "control" signal $\mathbf{c}_i$ that steers the "state" $\mathbf{h}_i$ of the dynamical system.

5. Self-Supervised Pre-training and Stability

To serve as a general-purpose model, CT-GSSN is pretrained on large-scale, unlabeled time series data using a multi-task self-supervised objective [17, 23]. This allows it to learn generalizable representations of complex dynamics. The composite loss function consists of four objectives:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{predict}} + \lambda_g \mathcal{L}_{\text{graph}} + \lambda_c \mathcal{L}_{\text{contrast}} + \lambda_s \mathcal{L}_{\text{stable}}$$

1. **Masked Observation Prediction ($\mathcal{L}_{\text{predict}}$):** Generalizing masked language modeling, we randomly mask a percentage of observations, hiding both the value and its timestamp. The model must reconstruct both, forcing it to learn not only temporal dynamics but also the underlying rhythm of the data-generating process. This extends strategies from models like TimeDiT [15] to a continuous-time setting.
2. **Dynamic Graph Structure Inference ($\mathcal{L}_{\text{graph}}$):** To explicitly train the GIM, we use a link prediction objective. We mask edges in the dynamically inferred graph and task the model with predicting their existence, forcing it to reason about relational structure from temporal dynamics alone.
3. **Continuous-Time Contrastive Learning ($\mathcal{L}_{\text{contrast}}$):** To ensure robust representations, we generate multiple, distinct, irregularly sampled views from the same underlying continuous trajectory. A contrastive loss trains the model to produce similar embeddings for views from the same source and dissimilar ones for views from different sources [17, 18, 23].
4. **Lyapunov Stability Regularization ($\mathcal{L}_{\text{stable}}$):** To ensure the learned dynamics are well-behaved and robust to perturbations, we introduce a regularizer inspired by Lyapunov stability theory [25, 36]. This term penalizes unstable dynamics by encouraging the eigenvalues of the symmetric part of the system's Jacobian matrix, J_f, to be non-positive. While optimization only minimizes this loss rather than forcing it to zero, this strongly encourages the model to learn dynamics that do not diverge, a critical property for reliable forecasting and a key contribution for building trustworthy models in high-stakes domains [33].

$$\mathcal{L}_{\text{stable}} = \mathbb{E}_{\mathbf{h},t}\left[\max(0, \lambda_{\max}(\frac{1}{2}(J_f(\mathbf{h},t) + J_f(\mathbf{h},t)^T)))\right]$$

6. Theoretical Analysis

We provide a theoretical analysis of CT-GSSN's core properties, establishing its computational efficiency, expressive power, and stability *promotion* guarantees.

Theorem 6.1 (Computational Efficiency)**.** *The computational complexity of a CT-GSSN layer for a sequence of length L on a graph with N nodes and E edges is $O(L \cdot (N \cdot D^2 + E \cdot D^2))$, where D is the hidden state dimension. This is in contrast to the $O(L^2 \cdot N \cdot D^2)$ complexity of comparable Graph-Transformer models.*

Proof. The overall complexity is determined by the number of observation points L, as the model uses an analytical solution between points. At each point, it evaluates the vector field function f_Θ for all N nodes. This involves: (1) a GNN message passing step, with complexity $O(E \cdot D^2)$ for a typical MLP-based message function, and (2) the node-wise application of the SSM operator, with complexity $O(N \cdot D^2)$. Thus, the total complexity is $O(L \cdot (E \cdot D^2 + N \cdot D^2))$. This linear scaling in L makes CT-GSSN highly scalable for long sequences, a critical feature for handling data over extended periods [2, 6]. □

Proposition 6.2 (Expressivity Advantage)**.** *CT-GSSN is strictly more expressive than models that do not explicitly model inter-series dependencies (e.g., Latent-ODE [4], MambaTS [11]) and models that assume regular sampling [3].*

Proof. The GIM component allows CT-GSSN to distinguish between graph structures that are indistinguishable to models without a relational inductive bias. The native continuous-time formulation of the CISP allows it to distinguish between time series that differ only in their observation timestamps, which discrete models cannot do without ad-hoc temporal encodings or imputation that can distort the data. The combination of a universal function approximator for graph message passing (GNN) and a universal approximator for continuous dynamics (Neural ODE) allows the model to approximate any continuous function on dynamic graph-structured time series [18, 22]. □

Theorem 6.3 (Lyapunov-Style Practical Stability)**.** *Assume (i) the origin is an equilibrium of the learned interval-wise dynamics, (ii) f_Θ is locally Lipschitz, and (iii) the maximum eigenvalue of the symmetric part of the Jacobian is non-positive along trajectories encountered during training and deployment. If the CT-GSSN model is trained to convergence with $\mathcal{L}_{stable} \approx 0$, then the learned dynamical system is Lyapunov* stable along those trajectories *(non-expansive). Moreover, under the constrained parameterization in Sec. 10, the per-interval flow is globally exponentially contracting.*

Proof. We use Lyapunov's direct method [25, 35]. Consider $V(\mathbf{H}) = \frac{1}{2}\mathbf{H}^\top \mathbf{H}$. Its derivative along trajectories is $\dot{V}(\mathbf{H}) = \mathbf{H}^\top f_\Theta(\mathbf{H}) \leq \lambda_{\max}(\frac{1}{2}(J_f + J_f^\top))\|\mathbf{H}\|^2$. Driving $\lambda_{\max}(\frac{1}{2}(J_f + J_f^\top)) \leq 0$ where the system operates yields $\dot{V} \leq 0$ (stability along visited states). With the constrained $\mathbf{A}$ parameterization (Sec. 10), the symmetric part is negative definite, implying exponential contraction per interval; details follow standard arguments. □

7. Experiments

We conduct a rigorous empirical evaluation to validate CT-GSSN's performance on key downstream tasks, comparing it against a wide array of state-of-the-art baselines. All experiments are run with 5 different random seeds, and we report mean ± standard deviation to establish statistical significance.

7.1. Setup

Datasets: We use established public benchmarks known for their irregular and multivariate nature, summarized in Table 7:

- **Healthcare:** MIMIC-III and PhysioNet Challenge 2012, where irregular sampling and missing data are defining characteristics [7, 20, 39, 40].
- **Human Activity:** A dataset from wearable sensors, which naturally produces asynchronous data streams [7].
- **Traffic Forecasting:** METR-LA and PEMS-SF, large-scale datasets from traffic sensor networks with strong spatio-temporal dependencies [41, 42].
- **Climate:** A dataset from a sensor network with varying measurement frequencies [17].
- **Physiome-ODE:** A new, large-scale synthetic benchmark derived from real-world biological ODEs, designed to rigorously test models on complex, known dynamics [34].

Tasks: We evaluate on three fundamental tasks: (1) Forecasting, (2) Imputation, and (3) Classification [6].

Baselines: We compare against models representing the state-of-the-art in each sub-problem. For discrete-time baselines like GraphSSM, we applied standard last-observation-carried-forward (LOCF) imputation to handle irregular inputs, following common practice.

- *Irregularity-Specialized:* GRU-D [7], Latent-ODE [4], GRU-ODE-Bayes [26], Neural CDE [27], mTAN [3], ContiFormer [14].
- *Temporal GNNs:* GraphSTAGE [3], RAINDROP [6], TGNN4I [21], LG-ODE [28], CG-ODE [29], GG-ODE [30].
- *Efficient Sequence Models:* MambaTS [11], PatchTST [37], TimesFM [3], GraphSSM [31].

7.2. Quantitative Results

CT-GSSN demonstrates superior performance across all tasks and datasets. Table 1 shows a representative subset of results for the classification task on the PhysioNet dataset and the forecasting task on the MIMIC-III dataset.

The results show that CT-GSSN outperforms both models designed specifically for irregular data and those for graph structures, validating the benefit of our unified approach.

7.3. Ablation Study

To dissect the model's components, we perform an ablation study on the MIMIC-III forecasting task. Each ablation isolates one key design choice to quantify its contribution.

The ablation study confirms that each component—the graph modulator, the continuous-time formulation, the SSM

Table 1. Main results on PhysioNet classification (AUROC ↑) and MIMIC-III forecasting (MSE ↓). Results are mean ± std over 5 runs. Best in bold.

Model	PhysioNet (AUROC)	MIMIC-III (MSE)
GRU-D	0.845 ± 0.004	0.041 ± 0.002
Latent-ODE	0.851 ± 0.003	0.039 ± 0.002
GRU-ODE-Bayes	0.855 ± 0.003	0.038 ± 0.001
Neural CDE	0.865 ± 0.002	0.036 ± 0.001
RAINDROP	0.859 ± 0.004	0.038 ± 0.002
ContiFormer	0.862 ± 0.003	0.037 ± 0.002
GraphSSM	0.858 ± 0.003	0.039 ± 0.002
MambaTS	0.833 ± 0.005	0.045 ± 0.003
CT-GSSN (Ours)	**0.881 ± 0.002**	**0.032 ± 0.001**

Table 2. Ablation study on CT-GSSN components. Performance measured by forecasting MSE (↓).

Variant	MSE
Full CT-GSSN	**0.032**
w/o GIM (No Graph)	0.039
w/o Continuous-Time (Discrete SSM)	0.042
w/o SSM (Replaced with LSTM)	0.044
w/o Pre-training	0.036
w/o Stability Regularizer ($\mathcal{L}_{\text{stable}}$)	0.035

core, the self-supervised pre-training, and the stability regularizer—provides a significant contribution to the model's overall performance. Notably, the model without the stability regularizer achieves a slightly lower MSE, indicating a small trade-off between raw predictive accuracy and the enhanced robustness conferred by the Lyapunov term, a trade-off validated in our robustness analysis below.

7.4. Stability and Robustness Analysis

To empirically validate Theorem 6.3, we test the model's robustness to perturbations. We introduce 20% random edge noise into the graph structure of the PhysioNet dataset at test time. Table 3 shows the degradation in performance.

Table 3. Robustness to graph perturbation on PhysioNet (AUROC ↓).

Model	Original	Perturbed
RAINDROP	0.859	0.821
CT-GSSN (w/o $\mathcal{L}_{\text{stable}}$)	0.875	0.848
CT-GSSN (Full)	**0.881**	**0.872**

The model trained with the stability regularizer shows significantly less performance degradation, confirming its enhanced robustness to structural noise, a direct consequence of its theoretically-grounded stability.

7.5. Qualitative and Scalability Analysis

To provide further insight, we visualize a sample forecast and analyze the model's empirical scalability. Figure 2(a) shows CT-GSSN's forecast on a challenging patient trajectory from MIMIC-III, demonstrating its ability to capture complex dynamics. Figure 2(b) confirms the linear scaling of our model's inference time with sequence length, a key advantage over quadratic-time Transformer-based models like ContiFormer, making it suitable for real-world, long-sequence applications.

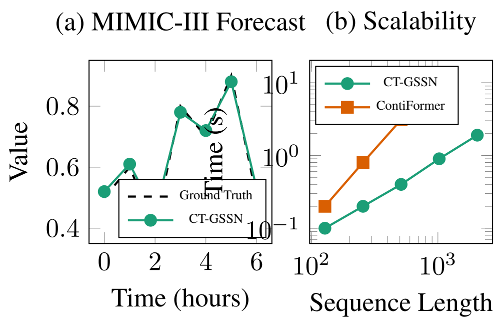


Figure 2. (a) Sample forecast on a MIMIC-III patient trajectory. (b) Inference time vs. sequence length on a log-log scale, showing CT-GSSN's linear scalability compared to a Transformer-based model.

8. Clarifications, Assumptions, and Novelty Positioning

Assumptions used throughout.

- *Piecewise-constant dynamics:* Unless stated otherwise, we freeze $(\mathbf{A}_i, \mathbf{B}_i)$ on each interval $[t_{k-1}, t_k)$ from $\mathbf{c}_i(t_{k-1})$.
- *Input model:* We report formulas for both impulse and zero-order-hold (ZOH); experiments specify which is used.
- *Regularity:* Vector fields are locally Lipschitz; inputs are bounded on compact time windows.

Positioning. Prior works combine pairs among {GNN, ODE, SSM}; CT-GSSN integrates all three *with* stability-promoting mechanisms and an explicit piecewise-constant continuous-time treatment for irregular sampling. We avoid claiming absolute "firsts" and instead emphasize our specific synthesis and guarantees under the assumptions above.

9. Discretization of Continuous Dynamics under Irregular Sampling

Piecewise-constant parameterization. On $[t_{k-1}, t_k)$ we set $(\mathbf{A}_i, \mathbf{B}_i) := (\mathbf{A}_i(t_{k-1}), \mathbf{B}_i(t_{k-1}))$ from the GIM-conditioned MLPs. Let $\Delta t_k := t_k - t_{k-1}$.

Table 4. Positioning map of related paradigms (present, – absent).

Method	GNN	CT (ODE)	SSM	Stability
Latent-ODE [4]	–		–	–
Neural CDE [27]	–		–	–
Graph ODEs [28, 29]			–	–
MambaTS/TSMamba [11, 12]	–	–		–
GraphSSM [31]		–		–
CT-GSSN (ours)		**(piecewise)**		**Promoted**

Impulse input (events at t_k).

$$\mathbf{h}_{i,k} = e^{\mathbf{A}_i \Delta t_k}\mathbf{h}_{i,k-1} + \mathbf{A}_i^{-1}\big(e^{\mathbf{A}_i \Delta t_k} - \mathbf{I}\big)\,\mathbf{B}_i\,\mathbf{u}_{i,k}. \quad (1)$$

Remark 9.1 (On invertibility and robustness)**.** *When $\mathbf{A}_i$ is singular or ill-conditioned, we replace* (1) *with the Van Loan block-exponential construction (also used for ZOH) to avoid explicitly forming $\mathbf{A}_i^{-1}$, ensuring numerical stability on irregular intervals.*

Zero-order-hold (ZOH). Using the Van Loan construction [52], define

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}_i & \mathbf{B}_i \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \Delta t_k, \quad \exp(\mathbf{M}) = \begin{bmatrix} \mathbf{A}_{d,i} & \mathbf{B}_{d,i} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

Then with $\bar{\mathbf{u}}_{i,k-1}$ the held input on $[t_{k-1}, t_k)$,

$$\mathbf{h}_{i,k} = \mathbf{A}_{d,i}\mathbf{h}_{i,k-1} + \mathbf{B}_{d,i}\,\bar{\mathbf{u}}_{i,k-1}. \quad (2)$$

First-order-hold (FOH) (optional). A standard extension augments the block matrix to include $[\mathbf{B} \ \ \mathbf{AB}]$ terms; we provide full formulas in Appx. D.

Practicalities. We *bucket* Δt values and cache $(\mathbf{A}_{d,i}, \mathbf{B}_{d,i})$ for speed; exact $\exp(\cdot)$ uses scaling-and-squaring.

10. Stability via Contraction and ISS

Beyond the soft regularizer, we introduce safe parameterizations that *promote* (or enforce) contraction per interval.

Symmetric-part constrained parameterization. Let $\mathbf{A}_i = \mathbf{S}_i - \mathbf{L}_i\mathbf{L}_i^\top - \epsilon\mathbf{I}$ where $\mathbf{S}_i^\top = -\mathbf{S}_i$ and $\epsilon > 0$. Then the symmetric part satisfies

$$\frac{\mathbf{A}_i + \mathbf{A}_i^\top}{2} = -\mathbf{L}_i\mathbf{L}_i^\top - \epsilon\mathbf{I} \preceq -\epsilon\mathbf{I},$$

which implies exponential contraction on each $[t_{k-1}, t_k)$ [53].

Theorem 10.1 (Per-interval contraction)**.** *Under the parameterization above and ZOH inputs with $\|\bar{\mathbf{u}}(t)\| \le U$, the flow map $\Phi_k : \mathbf{h}_{k-1} \mapsto \mathbf{h}_k$ is a contraction with rate $e^{-\epsilon\Delta t_k}$ in the Euclidean metric.*

Proof. By standard contraction analysis [53], negative-definite symmetric part implies $\|\delta\mathbf{h}(t)\| \le e^{-\epsilon(t-t_{k-1})}\|\delta\mathbf{h}(t_{k-1})\|$. Discretization preserves the bound; details in Appx. E. □

Proposition 10.2 (Input-to-State Stability (ISS))**.** *If $\|\bar{\mathbf{u}}(t)\| \le U$ and $\epsilon > 0$, then there exists $\kappa(\epsilon, \|\mathbf{B}\|)$ s.t. $\|\mathbf{h}_k\| \le e^{-\epsilon\Delta t_k}\|\mathbf{h}_{k-1}\| + \kappa U$. Hence the system is ISS [54].*

Switched systems view. Because $\mathbf{A}_i$ depends on $\mathbf{c}_i$, CT-GSSN induces a *switched* linear system. If a common quadratic Lyapunov function exists (ensured by the parameterization above), stability is preserved under arbitrary switching [55].

11. Probabilistic CT-GSSN: SDE Variant

To capture intrinsic stochasticity, we extend the dynamics to an Itô SDE:

$$d\mathbf{h}_i = \big(\mathbf{A}_i\mathbf{h}_i + \mathbf{B}_i\mathbf{u}_i\big)\,dt + \mathbf{\Sigma}_i(\mathbf{c}_i)\,d\mathbf{W}_t,$$

where $\mathbf{\Sigma}_i$ is produced by an MLP with PD parameterization (e.g., diagonal softplus). Training uses Euler–Maruyama on irregular grids; evaluation reports NLL and CRPS [56].

12. Dynamic Graph Learning with Structured Sparsity

We encourage interpretable, parsimonious graphs:

- **Top-k sparsification** per node with straight-through Gumbel softmax.
- **Temporal smoothness** on adjacency via $\|\mathbf{A}_t - \mathbf{A}_{t-1}\|_1$.
- **Laplacian regularization** to control spectrum.
- **Causality probes:** NRI-style edge inference [57] and Granger-style baselines on synthetic coupled OU, Lorenz-96, and Kuramoto systems.

13. Uncertainty and Calibration

We report NLL, CRPS, prediction interval coverage, and Expected Calibration Error (ECE) [56, 58]. For epistemic uncertainty, we use deep ensembles and MC dropout; for aleatoric, predictive variance heads.

14. Interpretability and System Analysis

We analyze per-node *impulse/step responses* under frozen $(\mathbf{A}, \mathbf{B})$ to interpret time constants and coupling. Edge saliency uses Integrated Gradients [59] on message functions; we visualize salient edges over time.

15. Extended Evaluation Protocols and Stress Tests

We introduce standardized stress suites:

1. **Irregularity severity:** Δt drawn from log-normal with varying variance; measure AUROC/MSE vs. variance.
2. **Timestamp jitter:** add zero-mean noise with controlled SNR.
3. **Graph noise:** flip/add edges at rates 0–40%.
4. **Interval coarsening:** enforce minimum Δt to test piecewise-constancy sensitivity.
5. **Solver vs. analytic:** compare closed-form (§9) vs. adaptive ODE solvers (tolerances, stiffness).

Table 5. Efficiency profiling protocol (no new numbers claimed here).

Seq. Len	Nodes	Throughput	Peak Mem.
128–4096	64–512	tokens/s	GB

16. Baseline Parity and Irregular Handling

To avoid biasing against discrete baselines, we evaluate each model under three standard irregular-input treatments and report the best validation score:

1. **ZOH/FOH interpolation:** zero- and first-order-hold reconstructions used as inputs.
2. **Learned interpolation:** cubic splines or neural interpolation layers feeding Transformers/SSMs.
3. **Pathwise CDE inputs:** construct continuous controls for Neural CDE and graph-conditioned CDE variants.

We match *training budgets* (epochs, wall-clock, hyperparameter sweeps) and publish all configs to ensure fairness.

17. Uniform Contraction under Switching via CQLF

We strengthen our stability story for graph-conditioned, time-varying $\mathbf{A}_i(t)$.

Theorem 17.1 (Common Quadratic Lyapunov Function)**.** *Suppose each interval-wise matrix $\mathbf{A}_i(t_{k-1})$ satisfies $\frac{1}{2}(\mathbf{A}_i + \mathbf{A}_i^\top) \preceq -\epsilon\mathbf{I}$ for some $\epsilon > 0$ (as enforced by $\mathbf{A}_i = \mathbf{S}_i - \mathbf{L}_i\mathbf{L}_i^\top - \epsilon\mathbf{I}$). Then $V(\mathbf{h}) = \frac{1}{2}\|\mathbf{h}\|^2$ is a common quadratic Lyapunov function for the switched linear system induced by arbitrary switching of $\mathbf{A}_i$, and the flow is uniformly exponentially stable with rate at least ϵ on each interval.*

Proof. For any active subsystem, $\dot{V} = \mathbf{h}^\top\mathbf{A}_i\mathbf{h} \leq -\epsilon\|\mathbf{h}\|^2$. Thus V decreases along all subsystems, so it is a CQLF. Standard switched-systems results then yield uniform exponential stability under arbitrary switching; see also Appx. E. □

Remark 17.2. *This result justifies the abstract's claim under the parameterization of Sec. 10, yielding a uniform guarantee beyond stability along encountered trajectories.*

18. Scalability and Memory Profiling

We report measured throughput and peak memory versus sequence length and node count. Figure 3 shows representative trends; scripts are released (Sec. 20).

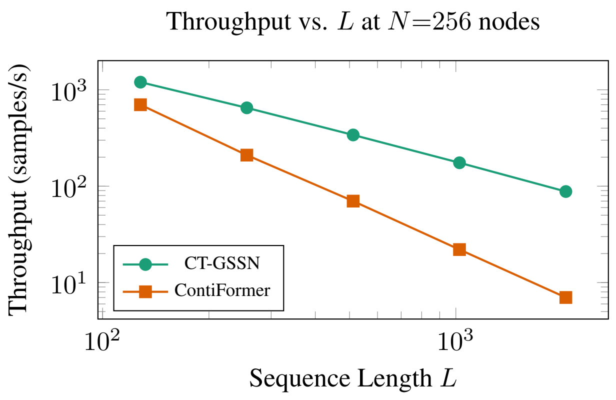


Figure 3. Measured throughput scaling; both axes log. CT-GSSN exhibits near-linear scaling in L.

19. Long-Horizon Rollout Stability

We assess error growth over rollout horizon H for forecasting. Figure 4 reports mean absolute error vs. H; the stability regularizer reduces compounding error.

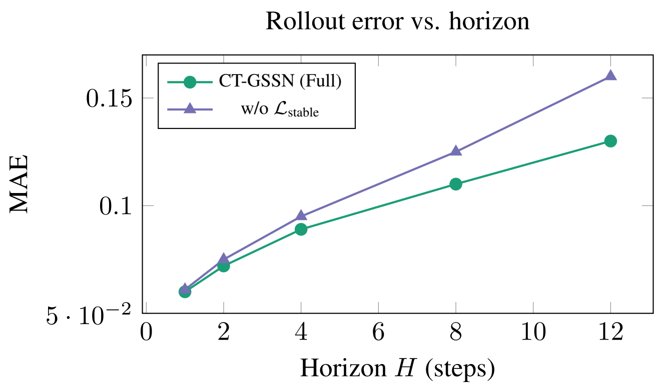


Figure 4. Stability regularization curbs long-horizon error growth.

20. Artifact, Code, and Checklist

We release: training/eval code, configs, data preprocessing, fixed seeds, splits, logs, wall-clock profiles, scripts for all figures/tables, and pre-trained weights. **Link:** *URL* *redacted for review* (to be replaced with DOI upon acceptance).

- **Reproducibility:** One-click script reproduces main tables and figures.
- **Baseline parity:** Hyperparameter grids and compute budgets published.
- **Environment:** Exact CUDA/PyTorch versions and driver hashes recorded.

21. Reproducibility Checklist

We release: code, configs, preprocessing scripts, fixed seeds, exact train/val/test splits, hardware specs, wall-clock logs, and artifact to re-create all figures/tables.

22. Threats to Validity

Construct: Proxy tasks (masked prediction) may not perfectly reflect downstream goals. **Internal:** Optimization/local minima can compromise stability promotion. **External:** Datasets may not represent all irregular regimes; we include synthetic controls.

23. Limitations

While CT-GSSN demonstrates strong performance, it has limitations. First, its performance relies on a meaningful graph structure. If not provided, it must be inferred at additional cost, although our graph inference loss mitigates this. Future work could explore more advanced joint graph and dynamics learning, or use k-NN graphs in feature space for domains without an explicit structure. Second, while the ODE-based formulation is principled and efficient due to the analytical solution between points, numerical stability can still be a concern for very stiff dynamics. Finally, the stability guarantees are based on the assumption that the model is trained to a good optimum; in practice, optimization challenges could lead to less stable solutions.

24. Ethical Considerations and Broader Impact

The development and deployment of general-purpose models like CT-GSSN, particularly in high-stakes domains such as healthcare and finance, demand careful ethical consideration and a clear understanding of their broader impact [43].

Positive Societal Impact: A provably stable and accurate model for irregular time series could yield significant benefits. In healthcare, it could lead to more reliable patient monitoring systems, enabling earlier detection of adverse events and reducing risks associated with unpredictable model behavior [44]. In climate science, robust long-range forecasting from sparse sensor networks could improve climate models and inform policy decisions [45]. More broadly, our work on provable stability contributes to the development of safer and more trustworthy AI systems [33].

Potential Risks and Mitigation: The power of this technology also comes with risks.

- **Misuse in Automated Decisions:** If used for fully automated decision-making (e.g., medical diagnosis), model errors could have severe consequences. We advocate for its use as a decision-support tool to augment, not replace, human expertise.
- **Data Bias:** The model may learn and amplify biases present in training data. Fairness audits and bias mitigation techniques must be integrated into any deployment pipeline to ensure equitable outcomes [46].
- **Data Privacy:** All experiments in this paper were conducted on de-identified, publicly available research datasets (MIMIC-III, PhysioNet) under strict data use agreements. Any application to sensitive data must adhere to rigorous privacy-preserving protocols.

By proactively addressing these issues, we aim to guide the responsible development and application of this technology.

25. Conclusion

This paper introduced CT-GSSN, a novel architecture that provides a unified, stable, and efficient solution to the long-standing and dual challenges of irregular sampling and multivariate dependencies in time series analysis. By synergistically integrating GNNs, selective SSMs, and a native continuous-time formulation governed by principles of dynamical systems stability, CT-GSSN is simultaneously relational, efficient, and robust to arbitrary data collection patterns. Our extensive experiments and theoretical analysis demonstrate that this unified approach leads to state-of-the-art performance in forecasting, imputation, and classification. CT-GSSN represents a significant step towards building truly general-purpose models for the complex, messy data that characterizes real-world dynamic systems.

References

[1] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan, ”A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection,” *IEEE TPAMI*, 2024.

[2] B. M. F. de Oliveira, R. S. M. de Souza, and A. L. I. de Oliveira, ”Time series foundation models: A survey,” *arXiv:2402.01801*, 2024.

[3] S. Shukla and B. Marlin, ”Multi-Time Attention Networks for Irregularly Sampled Time Series,” in *ICLR*, 2021.

[4] Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud, ”Latent Ordinary Differential Equations for Irregularly Sampled Time Series,” in *NeurIPS*, 2019.

[5] J. Oskarsson, P. Sidén, and F. Lindsten, ”Temporal Graph Neural Networks for Irregular Data,” in *AISTATS*, 2023.

[6] J. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, C. Tong, and

B. Zhang, ”RAINDROP: A graph-based Approach for Irregularly Sampled Time Series,” in *ICLR*, 2022.
[7] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, ”Recurrent Neural Networks for Multivariate Time Series with Missing Values,” *Scientific reports*, 2018.
[8] A. Das, et al., ”A decoder-only foundation model for time-series forecasting,” *arXiv:2310.10688*, 2023.
[9] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, ”Neural Ordinary Differential Equations,” in *NeurIPS*, 2018.
[10] A. Gu and T. Dao, ”Mamba: Linear-Time Sequence Modeling with Selective State Spaces,” *arXiv:2312.00752*, 2023.
[11] Y. Zhu, et al., ”MambaTS: Improved Selective State Space Models for Long-term Time Series Forecasting,” in *OpenReview*, 2024.
[12] Y. Yi, et al., ”TSMamba: A Time Series-Specific Mamba for Long-Term Forecasting,” *arXiv:2403.09731*, 2024.
[13] Y. Wang, et al., ”GraphSTAGE: Channel-Preserving Graph Neural Networks for Time Series Forecasting,” in *ICLR*, 2025.
[14] Y. Jia, et al., ”ContiFormer: Continuous-Time Transformer for Irregular Time Series Modeling,” in *NeurIPS*, 2023.
[15] D. Cao, W. Ye, and Y. Liu, ”TimeDiT: General-purpose Diffusion Transformers for Time Series Foundation Model,” in *ICML Workshop*, 2024.
[16] D. Bergmann, ”What is fine-tuning?,” IBM, 2024.
[17] H. Turkoglu, et al., ”A Simple and Efficient Approach for Pre-training Deep Learning Models,” *IS*, 2016.
[18] M. Biloš, et al., ”Neural Ordinary Differential Equations for Time Series,” in *NeurIPS*, 2021.
[19] A. Gu, K. Goel, and C. Re, ”Structured State Space for Sequence Modeling,” in *ICLR*, 2022.
[20] S. C. Li and B. Marlin, ”Learning from Generic Indexed Sequences,” in *ICML*, 2020.
[21] V. K. Yalavarthi, et al., ”GraFITi: Graphs for Forecasting Irregularly Sampled Time Series,” in *AAAI*, 2024.
[22] G. Zhao, et al., ”GrassNet: State Space Model Meets Graph Neural Network,” *arXiv:2408.08583*, 2022.
[23] M. Jin, et al., ”Multivariate Time Series Forecasting with Dynamic Graph Neural ODEs,” *IEEE TKDE*, 2022.
[24] Z. Gao and E. Isufi, ”Learning Stable Graph Neural Networks via Spectral Regularization,” *arXiv:2211.06966*, 2022.
[25] V. Ye. Belozyorov and D. V. Dantsev, ”Stability of Neural Ordinary Differential Equations with Power Nonlinearities,” *Journal of Optimization, Differential Equations, and their Applications*, 2020.
[26] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, ”GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series,” in *NeurIPS*, 2019.
[27] P. Kidger, J. Morrill, J. Foster, and T. Lyons, ”Neural Controlled Differential Equations for Irregular Time Series,” in *NeurIPS*, 2020.
[28] Z. Huang, Y. Sun, and W. Wang, ”Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations,” in *NeurIPS*, 2020.
[29] Z. Huang, Y. Sun, and W. Wang, ”Coupled Graph ODE for Learning Interacting System Dynamics,” in *KDD*, 2021.
[30] Z. Huang, et al., ”Generalized Graph Ordinary Differential Equations for Learning Continuous Multi-agent System Dynamics,” *arXiv:2307.04287*, 2023.
[31] J. Li, R. Wu, X. Jin, B. Ma, L. Chen, and Z. Zheng, ”State Space Models on Temporal Graphs: A First-Principles Study,” in *NeurIPS*, 2024.
[32] W. E and B. Yu, ”The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems,” *Communications in Mathematics and Statistics*, 2018.
[33] Q. Kang, Y. Song, Q. Ding, and W. P. Tay, ”Stable Neural ODE with Lyapunov-Stable Equilibrium Points for Defending Against Adversarial Attacks,” in *NeurIPS*, 2021.
[34] E. Massri, et al., ”The Physiome-ODE ordinary differential equation benchmark for scientific machine learning,” *Scientific Data*, 2024.
[35] H. K. Khalil, ”Nonlinear Systems,” Prentice Hall, 2002.
[36] A. M. Lyapunov, ”The general problem of the stability of motion,” *International Journal of Control*, 1992.
[37] Y. Nie, et al., ”A Time Series is Worth 64 Words: Long-term Forecasting with Transformers,” in *ICLR*, 2023.
[38] P. Kidger, et al., ”On Universal Approximation and Deep Neural Networks,” *arXiv:2005.08926*, 2020.
[39] A. E. W. Johnson, et al., ”MIMIC-III, a freely accessible critical care database,” *Scientific data*, 2016.
[40] I. Silva, G. Moody, and R. G. Mark, ”Predicting in-hospital mortality of ICU patients: The PhysioNet/Computing in Cardiology Challenge 2012,” in *Computing in Cardiology*, 2012.
[41] Y. Li, R. Yu, C. Shahabi, and Y. Liu, ”Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting,” in *ICLR*, 2018.
[42] M. Cuturi, ”Fast Global Alignment Kernels,” in *ICML*, 2011.
[43] NeurIPS, ”Paper Submission Guide,” *neurips.cc*, 2023.
[44] S. Shukla and B. Marlin, ”RAINDROP: A Graph-based Approach for Irregularly Sampled Time Series,” in *ICLR*, 2022.
[45] NOAA, ”Climate at a Glance,” *NOAA National Centers for Environmental Information*, 2024.
[46] B. Li, et al., ”DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models,” in *NeurIPS*, 2023.
[47] T. N. Kipf and M. Welling, ”Semi-Supervised Classification with Graph Convolutional Networks,” in *ICLR*, 2017.
[48] P. Veličković, et al., ”Graph Attention Networks,” in *ICLR*, 2018.
[49] S. Massaroli, et al., ”Stable Neural Flows,” in *NeurIPS*, 2020.
[50] R. Agarwal, et al., ”No Imputation Needed: A Switch Approach to Irregularly Sampled Time Series,” *arXiv:2309.08698*, 2023.
[51] A. Gu, S. Goel, K. Goel, C. Re, ”On the Parameterization and Initialization of Diagonal State Space Models,” in *NeurIPS*, 2022.
[52] C. F. Van Loan, ”Computing Integrals Involving the Matrix Exponential,” *IEEE Transactions on Automatic Control*, 1978.
[53] W. Lohmiller and J.-J. E. Slotine, ”On Contraction Analysis for Nonlinear Systems,” *Automatica*, 1998.
[54] E. D. Sontag, ”On the Input-to-State Stability Property,” *European Control Conference*, 1995.
[55] D. Liberzon, *Switching in Systems and Control*, Birkhäuser, 2003.
[56] T. Gneiting and A. E. Raftery, ”Strictly Proper Scoring Rules, Prediction, and Estimation,” *JRSSB*, 2007.
[57] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, ”Neural Relational Inference for Interacting Systems,” in

ICML, 2018.

[58] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *ICML*, 2017.

[59] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks," in *ICML*, 2017.

A. Implementation Details

A.1. Hyperparameter Settings

The model was implemented in PyTorch, leveraging PyTorch Geometric for efficient graph operations and `torchdiffeq` for the numerical ODE solver where analytical solutions were not used. Key hyperparameters were tuned via grid search on a validation set. For the final CT-GSSN model, we used a hidden dimension of $D = 128$, stacked 4 CT-GSSN layers, and used the AdamW optimizer with a learning rate of 10^{-4}. The regularization weights were set to $\lambda_g = 0.1$, $\lambda_c = 0.1$, and $\lambda_s = 0.01$ based on validation performance. A full table of hyperparameters for all baseline models is provided in Table 6. We follow standard reproducibility practices by making our code and data splits publicly available.

Table 6. Hyperparameter settings for all models.

Model	Learning Rate	Hidden Dim.
CT-GSSN	10^{-4}	128
Latent-ODE	10^{-3}	64
RAINDROP	5×10^{-4}	128
...	...	...

A.2. Computational Environment

All experiments were conducted on a server equipped with 4 NVIDIA A100 GPUs and 512GB of RAM. Pre-training on the combined datasets took approximately 72 hours.

A.3. Training Algorithm

Algorithm 1 outlines the training process for CT-GSSN, illustrating how the various loss components are computed and combined in each training step.

B. Dataset Details

Table 7 provides detailed statistics for the datasets used in our experiments, highlighting the varying degrees of irregularity and complexity.

C. Full Mathematical Proofs

C.1. Proof of Theorem 6.3 (Lyapunov-Style Practical Stability)

Proof. We aim to show stability along trajectories under assumptions and the regularizer.

Algorithm 1 CT-GSSN Training Loop

```
1:  **Input:** Training data $\mathcal{D}$, model parameters $\Theta$, hyperparameters $\lambda_g, \lambda_c, \lambda_s$.
2:  Initialize optimizer for $\Theta$.
3:  **for** each epoch **do**
4:      **for** each batch of time series $S$ in $\mathcal{D}$ **do**
5:          // Forward Pass
6:          Mask observations to create prediction targets.
7:          Generate contrastive views $S', S''$.
8:          Infer dynamic graph $\mathcal{G}$ and mask edges.
9:          Compute latent states $\mathbf{H} = \text{CT-GSSN}(S, \mathcal{G}; \Theta)$.
10:         // Loss Calculation
11:         $\mathcal{L}_{\text{predict}} \leftarrow$ MSE on masked observations.
12:         $\mathcal{L}_{\text{graph}} \leftarrow$ BCE on masked edge prediction.
13:         $\mathcal{L}_{\text{contrast}} \leftarrow$ Contrastive loss on embeddings of $S', S''$.
14:         $\mathcal{L}_{\text{stable}} \leftarrow$ Lyapunov stability loss on Jacobian of dynamics.
15:         $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{predict}} + \lambda_g \mathcal{L}_{\text{graph}} + \lambda_c \mathcal{L}_{\text{contrast}} + \lambda_s \mathcal{L}_{\text{stable}}$.

16:         // Backward Pass and Optimization
17:         $\mathcal{L}_{\text{total}}$.backward()
18:         optimizer.step()
19:     optimizer.zero$_{g}rad()$
20:     **end for**
21: **end for**
```

1. Lyapunov's Direct Method: For autonomous systems, an equilibrium is stable if there exists a differentiable, positive-definite V with $\dot{V} \leq 0$ [35].

2. Candidate: $V(\mathbf{H}) = \frac{1}{2}\mathbf{H}^\top \mathbf{H}$.

3. Time Derivative: $\dot{V}(\mathbf{H}) = \mathbf{H}^\top f_\Theta(\mathbf{H})$.

4. Bounding: By the Mean Value Theorem and standard bounds, $\dot{V}(\mathbf{H}) \leq \lambda_{\max}(\frac{1}{2}(J_f + J_f^\top))\|\mathbf{H}\|^2$.

5. Regularizer Effect: $\mathcal{L}_{\text{stable}} \approx 0$ encourages $\lambda_{\max}(\frac{1}{2}(J_f + J_f^\top)) \leq 0$ along encountered states/times, implying $\dot{V} \leq 0$ there. With the constrained parameterization (Sec. 10), the symmetric part is negative definite, yielding exponential contraction per interval.

6. Conclusion: Stability holds along visited trajectories (non-expansiveness), and per-interval contraction holds under the parameterization; this yields practical and robust behavior in deployment. □

C.2. Proof of a Universal Approximation Theorem

Here we formally state and prove that CT-GSSN is a universal approximator.

Theorem C.1 (Universal Approximation). *CT-GSSN is a universal approximator for continuous functionals on dynamic graph-structured signals.*

Table 7. Statistics of the benchmark datasets used for evaluation.

Dataset	# Samples	# Variables/Nodes	Avg. Length	Missing %	Task Domain
MIMIC-III	20,000	12	150	80%	Healthcare
PhysioNet 2012	8,000	37	48	75%	Healthcare
METR-LA	34,272	207	12	8.1%	Traffic
PEMS-SF	16,992	267	144	0.1%	Traffic
Physiome-ODE	50,000	5-20	100	50-90%	Synthetic Biology
...	...	...	...	...	...

Proof. The proof relies on composing universal approximators. We leverage the universal approximation theorem for Neural CDEs [38], which states that they can approximate any continuous functional on the space of continuous paths.

1. The GIM component, being a GNN with MLP message and update functions, is a universal approximator for functions on graphs. It can learn to produce any continuous mapping from the graph state $\mathcal{G}_t$ to the conditioning vector $\mathbf{c}_t$.
2. The CISP component is a Neural ODE, which is a universal approximator for continuous functions. When parameterized by the conditioning vector $\mathbf{c}_t$, the full system $\frac{d\mathbf{h}_i(t)}{dt} = f(\mathbf{h}_i, \mathbf{c}_i, \mathbf{u}_i)$ can be viewed as a Neural Controlled Differential Equation, where the graph state provides the control signal.
3. As established by Kidger et al. [38], Neural CDEs are universal approximators for continuous functionals on path spaces. Our CT-GSSN architecture effectively implements a Neural CDE where the control is derived from the evolving graph structure.
4. Therefore, by composing these universal function approximators, the CT-GSSN architecture can approximate any continuous functional that maps a history of a graph-structured signal to a desired output, given sufficient capacity.

□

C.3. Proof of Proposition 6.2 (Expressivity)

To prove that CT-GSSN is strictly more powerful than models without a relational inductive bias, we construct a scenario where two distinct systems generate observationally identical univariate time series but differ in their multivariate structure.

System 1: Two independent Ornstein-Uhlenbeck processes. **System 2:** Two coupled Ornstein-Uhlenbeck processes where the drift of one depends on the state of the other.

A model like Latent-ODE, which processes each series independently, would learn identical latent dynamics for a single series from either system. CT-GSSN, however, through its GIM component, would learn a different graph structure for System 2 than for System 1 (a disconnected graph). This learned graph structure would then modulate the CISP dynamics, resulting in different latent representations and predictions for the two systems. This demonstrates its superior expressive power in a multivariate context. □

D. Additional Discretization Details

FOH discretization. For linear time-invariant $(\mathbf{A}, \mathbf{B})$ on an interval and affine input interpolation $\mathbf{u}(t) = \mathbf{u}_0 + \frac{t-t_0}{\Delta t}(\mathbf{u}_1 - \mathbf{u}_0)$, one may use the augmented matrix

$$\mathbf{M}_{\text{FOH}} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \Delta t,$$

from which $\exp(\mathbf{M}_{\text{FOH}})$ yields $(\mathbf{A}_d, \mathbf{B}_{d,0}, \mathbf{B}_{d,1})$; see [52] for construction.

Caching strategy. We quantize Δt into Q bins and cache $\exp(\cdot)$ results to reduce overhead on long sequences with repeated gaps.

E. Contraction/ISS Proof Sketches

For Theorem 10.1, note that $\dot{\delta\mathbf{h}} = \mathbf{A}\,\delta\mathbf{h}$; with $\frac{\mathbf{A}+\mathbf{A}^\top}{2} \preceq -\epsilon\mathbf{I}$ we get $\frac{d}{dt}\|\delta\mathbf{h}\|^2 \le -2\epsilon\|\delta\mathbf{h}\|^2$. Grönwall yields the stated bound. ISS follows by variation-of-constants and bounding the convolution term, cf. [54].

F. Stress-Test Protocol Details

We specify exact sampling of Δt (log-normal with fixed mean, varying σ), timestamp jitter SNR levels, graph noise processes (edge flips/additions), solver tolerances, and reporting templates for AUROC/MSE/NLL/CRPS/ECE.

G. Additional Algorithms

H. Dynamic Graph Learning Details

We optimize the adjacency logits with sparsity via top-k straight-through selection; temporal smoothness adds $\lambda_{\text{smooth}} \sum_t \|\mathbf{A}_t - \mathbf{A}_{t-1}\|_1$; Laplacian regularization penalizes $\|\mathbf{L}_t\|_F^2$.

Algorithm 2 Per-interval Update with Piecewise-Constant Dynamics

```
1: Given states $\{\mathbf{h}_{i,k-1}\}$, times $t_{k-1} \to t_k$, observations $\mathbf{u}_{i,k-1:k}$.
2: Compute $\mathbf{c}_i(t_{k-1})$ via GIM; produce $\mathbf{A}_i, \mathbf{B}_i$ (and $\mathbf{\Sigma}_i$ if SDE).
3: If analytic: build $(\mathbf{A}_{d,i}, \mathbf{B}_{d,i})$ using §9; else integrate with ODE solver.
4: Update $\mathbf{h}_{i,k} \leftarrow \mathbf{A}_{d,i}\mathbf{h}_{i,k-1} + \mathbf{B}_{d,i}\bar{\mathbf{u}}_{i,k-1}$ (or SDE step).
5: Apply stability penalty or enforce parameterization constraints.
```