Probabilistic Learning of Multivariate Time Series with Temporal Irregularity

Yijun LI, Cheuk Hang LEUNG, Qi WU

***Abstract*—Probabilistic forecasting of multivariate time series is essential for various downstream tasks. Most existing approaches rely on the sequences being uniformly spaced and aligned across all variables. However, real-world multivariate time series often suffer from temporal irregularities, including nonuniform intervals and misaligned variables, which pose significant challenges for accurate forecasting. To address these challenges, we propose an end-to-end framework that models temporal irregularities while capturing the joint distribution of variables at arbitrary continuous-time points. Specifically, we introduce a dynamic conditional continuous normalizing flow to model data distributions in a non-parametric manner, accommodating the complex, non-Gaussian characteristics commonly found in real-world datasets. Then, by leveraging a carefully factorized log-likelihood objective, our approach captures both temporal and cross-sectional dependencies efficiently. Extensive experiments on a range of real-world datasets demonstrate the superiority and adaptability of our method compared to existing approaches. The data and code supporting this work are available at https://github.com/lyjsilence/RFN.**

***Index Terms*—probabilistic forecasting, multivariate time series, irregular sampling, recurrent neural networks, normalizing flow models, neural ODEs**

I. Introduction

MULTIVARIATE time series (MTS) data, where multiple variables are recorded and evolve over time, are essential across fields such as healthcare, finance, and climate science. For instance, in healthcare, MTS data can track a patient's vital signs, enabling early detection of potential health issues. In finance, it supports forecasting market behaviors and asset correlations, which is essential for risk management and investment strategies. Probabilistic forecasting of MTS is indispensable, as it supports downstream tasks like anomaly detection, risk assessment, and decision-making. This involves predicting not only central measures like the mean or median but also quantiles and confidence intervals that are essential in high-stakes decision-making scenarios [1].

At the variable level, MTS data exhibit serial dependence, meaning the marginal distributions of individual variables at different time points are not independent [2]. At the group level, the dependence structure encompasses relationships like the copula function of the joint distribution, where components may be strongly interdependent. Both variable-level serial dependencies and group-level interdependencies can vary over time. Probabilistic forecasting entails specifying the joint distribution of variables in MTS, learning its representation from data, and predicting its evolution over time [3].

However, real-world MTS data rarely follow regular sampling intervals. Instead, these datasets often exhibit temporal irregularities, creating multivariate irregular time series where observations are separated by uneven time intervals. Temporal irregularities occur frequently in various applications. For example, climate data might have gaps if monitoring equipment fails to record at scheduled times. Patient measurements may be taken at inconsistent times, resulting in irregular sampling.

These irregularities, which may be random or due to specific factors, present significant challenges for probabilistic forecasting in MTS [4, 5]. Irregular sampling disrupts the modeling of serial dependencies, as traditional methods rely on evenly spaced data points to capture temporal patterns accurately. Additionally, it complicates the modeling of interdependencies among variables, as unaligned time points make it harder to reveal important relationships within the data. This irregularity increases the difficulty of accurately characterizing the joint distribution, adding uncertainty to the forecasting process [6]. Before presenting our approach, we review existing methods for handling temporal irregularity and modeling joint distributions.

A. On Handling Temporal Irregularity

There are three primary approaches to handling irregular sampling. The first approach involves converting an irregularly sampled time series into one with evenly spaced time intervals before making predictions. The discretization method selects a larger uniformly spaced time interval, with each interval containing several observations, and computes the mean of these observations to represent the interval's value. However, this method loses local information due to averaging. In contrast, the imputation method interpolates the missing values of the lower-frequency variables instead of averaging the higher-frequency variables. It keeps the local information of the higher-frequency variables intact and uses models such as the Gaussian Process regression model [7], the Recurrent Neural Networks (RNNs) [8], and the Generative Adversarial Networks (GANs) [9] to impute the missing values of the lower-frequency components.

The next approach proposes using end-to-end models to avoid the "interpolate first and predict later" idea. This approach modifies classical recurrent architectures to encode the information embedded in irregular temporal patterns. For example, Che et al. [10] added an exponential decay mechanism in the hidden state. Neil et al. [11] extended the Long

The first two authors contributed equally.
Corresponding author: Qi WU (qiwu55@cityu.edu.hk)
The authors are with the School of Data Science, City University of Hong Kong, Hong Kong.
Manuscript received April, 2024;

short-term memory (LSTM) unit by adding a new time gate controlled by a parametrized oscillation with a frequency range. Additionally, Mozer et al. [12] incorporated multiple time scales of the hidden state and made a context-dependent selection of time scales for information storage and retrieval.

At last, Chen et al. [13] introduced the Neural ODE framework by extending discrete neural networks into continuous-time networks, which makes it a natural candidate for handling data with arbitrarily small time intervals. For instance, Rubanova et al. [14] proposed the Latent ODE and were the first to embed Neural ODEs in a Variational Autoencoder [15] to address the problem of irregularly sampled time series. De Brouwer et al. [16] integrated the Neural ODEs in the classical Gated Recurrent Unit (GRU) cell and derived the dynamics of the hidden state. Unlike the classical GRU cell, which keeps the hidden state constant in the absence of observations, the continuous-time GRU cell learns to evolve the hidden state using Neural ODEs. Building upon similar ideas, Lechner et al. [17] transformed the standard LSTM into a continuous version to address the issues of gradient vanishing and exploding. In section II-A, we will provide more background on Neural ODEs and how to utilize them to model irregularly sampled data.

B. On Modeling Joint Data Distribution

For point estimation tasks, vanilla recurrent architectures, including RNN, GRU, and LSTM, can capture different aspects of the aforementioned properties of the MTS data. However, they are not directly applicable to distribution prediction due to the deterministic nature of the transition functions of their hidden states, which do not account for modeling uncertainties [18].

One class of models modifies the output function of neural networks to model the joint distribution or quantile function. For example, the models in [16, 19, 20] assume the data-generating process follows parametric distribution, such as the multivariate Gaussian (for continuous variables) and multivariate Negative Binomial (for discrete variables). Alternatively, researchers use quantile regression to fit the quantile function of the joint distribution. They use the quantile loss [21, 22] or the continuous ranked probability score [23] as the objective function to train the model and predict multiple quantile points simultaneously conditional on the hidden states.

Recently, unsupervised deep learning models have been utilized to learn the joint distribution of data, including integrating variational autoencoders [18], normalizing flows [24, 25], or diffusion models [26] into RNNs. Among these, the flow models are flexible in capturing intricate and evolving dependence structures and impose no assumptions about the functional form of the joint data distribution. These characteristics make them attractive for dealing with complex data, although they do not specifically address the structural aspects of irregular sampling. In section II-B, we shall further detail the background of representing data distribution using the normalizing flow approach.

C. Our Approach and Contributions

Discussions in I-A and I-B unveil dislocations and disparities among ideas of handling temporal irregularities in the data and ideas to model its joint distribution. This paper bridges this gap by introducing a deep learning solution called the *Recurrent Flow Network* (RFN). It can seamlessly integrate the treatment of temporal irregularities with the learning of joint data distribution. Its novelties are as follows.

(i) The proposed RFN framework formulates a two-layer representation that distinguishes marginal learning of variable dynamics from multivariate learning of joint data distribution. It is a versatile methodology that can be trained end-to-end and accommodates synchronous and asynchronous data structures. It also broadly applies to underlying recurrent architectures. Once the joint data distribution is learned, it is ready for sampling despite the distribution being non-parametrically represented via neural networks.

(ii) The joint learning layer resolves the struggle faced by existing models [16, 19, 20] in achieving a non-parametric representation of non-Gaussian data distribution, simultaneously with a flexible choice of information to generate time variation. The conditional CNF (Continuous Normalizing Flow) representation we developed enables one to choose what information to use to drive the time variation of the base distribution and the flow map without compromising any non-parametric capacity to represent the non-Gaussian data distribution.

(iii) Building upon (ii), we strategically condition the log-likelihood objective on the observation times. This conditioning structure enables the optimizer to fully acknowledge and account for both the uneven spacing aspect and the asynchrony aspect of temporal irregularity in the MTS data. By conditioning on the observation times, the RFN ensures that the model incorporates the specific time points at which the data is observed.

We validate the novelties mentioned above through synthetic experiments and demonstrate the overall performance of the RFNs on three real-world datasets. The synthetic studies simulate sample paths of a multivariate correlated Geometric Brownian Motion process to verify the ability of our approach to capture the conditional joint distribution. Meanwhile, the experimental datasets include the physical activities of the human body from the MuJoCo module [14], the climate records of weather from the USHCN dataset [27], and the minute-level transaction records of eight stocks in the biotechnology sector of NASDAQ market [28]. We compare four baseline models in terms of their performance in vanilla forms and the performances utilizing the RFN specification. The results show that the RFN framework has broad applicability and significantly improves existing approaches.

II. Backgrounds

Understanding the working mechanism of the proposed RFN model requires the knowledge of Neural ODEs, their applications to model irregularly sampled data, and the flow representation of distributions. This section summarizes these subjects to make the paper self-contained. Throughout the paper, we denote random variables as follows: X for scalar,

X for vector, and $\mathbf{X}$ for matrix. Their corresponding sample values are denoted as x, x, and $\mathbf{x}$ accordingly.

A. Neural Ordinary Differential Equations

Neural ODEs were developed as the continuous limit of the ResNet model. The ResNet model solves the degradation problem of neural networks where researchers noticed that, as the network layers go deeper, the training loss begins to increase steadily once the network depth crosses a certain threshold [29]. Consider a L-layer network, with x_0 being the input, x_l being the output of each layer $l \in \{1, \cdots, L\}$, and $f_{\theta_l}(\cdot)$ being the learning functions of layer l. Instead of learning the mapping from x_0 to x_L directly, ResNet learns the difference between the input and output of each layer:

$$\mathrm{x}_l = \mathrm{x}_{l-1} + \mathrm{f}_{\theta_l}(\mathrm{x}_{l-1}). \tag{1}$$

Chen et al. [13] proposed that taking the limit of the number of layers to infinity shall turn discrete layers into continuous layers. The resulting continuous limit of the recursive equation (1) is an ODE:

$$\frac{d\mathrm{x}(l)}{dl} = \mathrm{f}_\theta(\mathrm{x}(l), l). \tag{2}$$

Solving (2) with initial condition $\mathrm{x}(0)$ is equivalent to the forward pass of ResNet. One can use numerical methods such as the Euler and the Runge–Kutta methods to solve (2).

B. Unconditional Normalizing Flow

For tasks related to probabilistic forecasts, one needs a representation of the data distribution. Let $p(\mathrm{x})$, $\mathrm{x} \in \mathbb{R}^D$ be the probability density of the data-generating distribution and $p(\mathrm{z})$, $\mathrm{z} \in \mathbb{R}^D$ be the probability density of the base distribution which is typically set as the standard normal, i.e., $\mathrm{Z} \sim \mathcal{N}(0, \mathbb{I}_D)$. The idea of the normalizing flow model is to find a differentiable bijective function $\mathrm{f} = [f^1, \cdots, f^D]^\top$ which can map samples from Z to X [30, 31]:

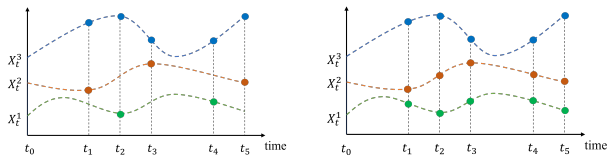
$$\mathrm{f} : \mathbb{R}^D \to \mathbb{R}^D; \quad \mathrm{f}(\mathrm{z}) = \mathrm{x}.$$

In the discrete formulation, f is typically specified as a sequence of neural networks, $\mathrm{f} = \mathrm{f}_1 \circ \cdots \circ \mathrm{f}_{M-1} \circ \mathrm{f}_M$. However, designing the architectures of $\mathrm{f}_1, \cdots, \mathrm{f}_M$ is challenging because they need to satisfy three conditions: being bijective, differentiable, and facilitating the computation of the determinant of the Jacobian of the function f.

The continuous normalizing flow model (CNF) [32] offers a solution to this challenge by extending the composition of discrete maps into a continuous map, whose differential form reads as follows:

$$\begin{aligned} &\frac{\partial \mathrm{z}(s)}{\partial s} = \mathrm{f}(\mathrm{z}(s), s; \theta), \quad s \in [s_0, s_1], \\ &\text{where} \quad \mathrm{z}(s)|_{s=s_0} = \mathrm{z}, \ \mathrm{z}(s)|_{s=s_1} = \mathrm{x}. \end{aligned} \tag{3a}$$

Unlike the physical time t, s is called the *flow time* of the dynamics (3a). At the initial flow time s_0, the value of the flow $\mathrm{z}(s_0)$ is set as z, which samples from the base distribution $p(\mathrm{z})$ of the base random variable Z. At the terminal flow time s_1, the value of the flow $\mathrm{z}(s_1)$ is set to equal x, which is the observed sample from the distribution of the true data-generating distribution $p(\mathrm{x})$.



(a) Syn-MTS (b) Asyn-MTS

Fig. 1. (a) and (b) are examples of Syn-MTS and Asyn-MTS where observed data points are marked as solid circle dots. While the time intervals between consecutive observation times are unevenly spaced in both cases, component observations of the Syn-MTS sample path are always aligned. In contrast, in the Asyn-MTS case, no observation time has complete observations. This demonstrates that uneven spacing originates at the univariate level, while asynchrony arises exclusively in the multivariate context.

The discrete formulation requires careful design of the weight matrices of f_j, $1 \le j \le M$, to be triangular to facilitate computing the Jacobian's determinant easily. However, the computation of the Jacobian determinant is replaced with relatively cheap trace operations thanks to the Instantaneous Change of Variables theorem [13] in the continuous formulation. Consequently, the log-density of the continuous flow follows the following equation:

$$\begin{aligned} &\frac{\partial \log p(\mathrm{z}(s))}{\partial s} = -\operatorname{Tr}\left[\partial_{\mathrm{z}(s)} \mathrm{f}\right], \quad s \in [s_0, s_1], \\ &\text{where} \quad p(\mathrm{z}(s))|_{s=s_0} = p(\mathrm{z}), \ p(\mathrm{z}(s))|_{s=s_1} = p(\mathrm{x}). \end{aligned} \tag{3b}$$

Solving equations (3a) and (3b) together, we have

$$\begin{bmatrix} \mathrm{x} \\ \log p(\mathrm{x}) \end{bmatrix} = \begin{bmatrix} \mathrm{z} \\ \log p(\mathrm{z}) \end{bmatrix} + \int_{s_0}^{s_1} \begin{bmatrix} \mathrm{f}(\mathrm{z}(s), s; \theta) \\ -\operatorname{Tr}\left[\partial_{\mathrm{z}(s)} \mathrm{f}\right] \end{bmatrix} ds. \tag{3}$$

III. Data Structure & Problem Statement

MTS are sequences of data where multiple variables are observed over time. Each variable may exhibit dependencies on both its own past values and the past values of other variables. Consider a MTS dataset containing N instances. Each instance is a D-dimensional sample path. All instances span the same $[0, T]$ period. For example, this dataset could represent climate recordings, consisting of N daily records, each spanning 24 (T) hours. During each day, multiple indices (D), such as temperature and precipitation, are observed. However, some recorded values may be missing due to various factors, such as equipment failure.

To account for the presence of temporal irregularity in a given instance $i \in \{1, \cdots, N\}$, we first collect all time points at which at least one variable has an observation and define this collection as the time vector of observations:

$$\mathrm{t}^i := [t_1^i, \cdots, t_{K_i}^i], \quad 0 \le t_1^i \le \cdots \le t_{K_i}^i \le T.$$

At a particular observation time $t \in \mathrm{t}^i$, we use $\mathrm{x}_{i,t} \in \mathbb{R}^{D \times 1}$ to denote the time-t sample values of the random vector X_t of the i^{th} instance, within which we use $x_{i,t}^d \in \mathbb{R}$ to denote the d^{th} component, which is the sample value of the d^{th} random scalar X_t^d, We also set $x_{i,t}^d = 0$ if no observation for the d^{th} at time t. Thus, we have

$$\mathrm{x}_{i,t} := [x_{i,t}^1, \cdots, x_{i,t}^d, \cdots, x_{i,t}^D]^\top, \quad \text{where } t \in \mathrm{t}^i.$$

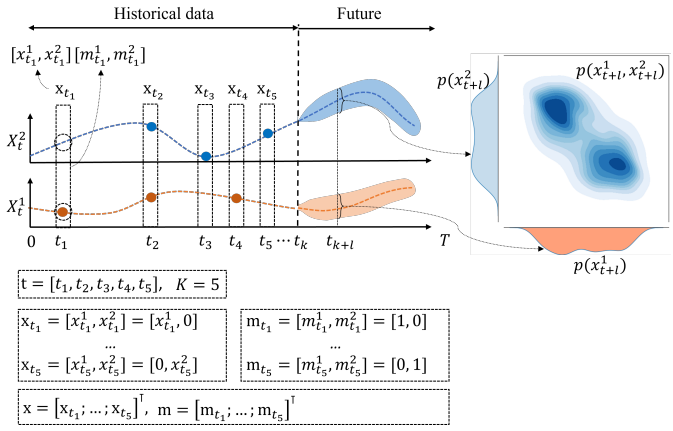


Fig. 2. The data structure and notations for one instance.

We then aggregate $\mathrm{x}_{i,t}$ from all observation times $t = t^i_1, t^i_2, \cdots, t^i_{K_i}$ of the instance i to form the instance-level data matrix $\mathbf{x}_i \in \mathbb{R}^{D \times K_i}$, where

$$\mathbf{x}_i := [\mathrm{x}_{i,t^i_1}; \mathrm{x}_{i,t^i_2}; \cdots; \mathrm{x}_{i,t^i_{K_i}}]^\top, \quad i \in [1, \cdots, N].$$

Finally, the entire MTS dataset is the collection of all N instances $\{\{\mathbf{x}_1\}; \{\mathbf{x}_2\}; \cdots; \{\mathbf{x}_N\}\}$.

An instance $\mathbf{x}_i$ can be synchronous or asynchronous depending on whether $x^d_{i,t}$ is observed or not for any combination of the variable dimension $d \in \{1, \cdots, D\}$ and the observation time $t \in \mathrm{t}^i$.

Definition 1. ***Synchronous multivariate time series (Syn-MTS)****: An instance $\mathbf{x}_i$ where all of its D component series have observations at each and every time points $t \in \mathrm{t}^i$ (see Fig. 1(a)).*

Definition 2. ***Asynchronous multivariate time series (Asyn-MTS)****: An instance $\mathbf{x}_i$ where at least one of its D component series does not have all observations at all time point $t \in \mathrm{t}^i$ (see Fig. 1(b)).*

To precisely distinguish between Asyn-MTS and Syn-MTS, we can use the mask matrix. For each instance i with corresponding data matrix $\mathbf{x}_i$, its mask matrix is $\mathbf{m}_i$ such that

$$\mathbf{m}_i := [\mathrm{m}_{i,t^i_1}; \mathrm{m}_{i,t^i_2}; \cdots; \mathrm{m}_{i,t^i_{K_i}}]^\top.$$

Here, $\mathrm{m}_{i,t}$ is a vector that denotes whether the constituent component variables are observed at time $t \in \mathrm{t}^i$, i.e.,

$$\mathrm{m}_{i,t} := [m^1_{i,t}, \cdots, m^d_{i,t}, \cdots, m^D_{i,t}]^\top, \quad \text{where}$$
$$m^d_{i,t} = \begin{cases} 1, & \text{if } x^d_{i,t} \text{ is observed;} \\ 0, & \text{if } x^d_{i,t} \text{ is unobserved.} \end{cases}$$

In the sequel, we shall drop the instance script i to lighten notations, e.g., t^i_k, $\mathrm{x}_{i,t}$ and $\mathrm{m}_{i,t}$ shall become t_k, x_t and m_t, whenever the context is clear. In Fig. 2, we give a plot explanation of an instance with two variables.

Problem Statement. Given the historical value of Syn-MTS or Asyn-MTS observations $\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_k}$, the objective of our work is to learn a non-linear mapping function to estimate the joint distribution $p(\mathrm{x}_{t_{k+l}})$ at arbitrary future time t_{k+l} in a continuous-time manner. In particular, the joint distribution can be decomposed into marginal distributions $p(x^1_{t_{k+l}}), \cdots, p(x^D_{t_{k+l}})$, which allows for interval estimation of each individual variable. In Fig. 2, we provide a graphical representation of the goal we aim to achieve, which demonstrates the probabilistic forecasting of two irregularly sampled and non-Gaussian variables over time.

IV. Model Framework

In this section, we introduce our RFN model framework, designed for probabilistic forecasting of multivariate irregular time series in both synchronous and asynchronous cases. A visual representation is provided in Figure 3.

The RFN comprises two main layers: the marginal learning layer and the joint learning layer. In the marginal learning layer, the multivariate irregular time series is processed using any advanced sequential model suitable for such data. The resulting representation, the hidden state of sequential models, is then passed to the joint learning layer.

The joint learning layer aims to learn the unknown data distribution using the conditional CNF model, leveraging the change of variable theorem. Specifically, conditioned on the hidden state, we learn a differentiable bijective function that maps the unknown distribution to a simple base distribution—such as a multivariate normal distribution—for which the likelihood is easy to compute. The likelihood of a real data point in the unknown distribution can then be calculated by combining the likelihood of the transformed data point under the base distribution with the transformation loss.

Furthermore, the joint learning layer is tailored to handle both Syn-MTS and Asyn-MTS scenarios. The key difference between these cases is that in Asyn-MTS, some variables may be missing at certain time points, making it impossible to compute the likelihood. To address this issue, we force each variable in the base distribution to be independent and compute the likelihood of the observed variables only.

Below, we first describe the marginal learning layer, followed by a detailed explanation of the joint learning layer for both synchronous and asynchronous cases.

A. Marginal Learning Layer

The marginal learning layer aims to acknowledge temporal irregularities in the MTS data. Various existing recurrent architectures targeting handling irregularly sampled time series can be employed, such as those discussed in Section I-A. In this section, we dedicate to providing one example, GRU-ODE-Bayes [16], to elucidate the functionality of the marginal learning layer and its interactions with the joint learning layer.

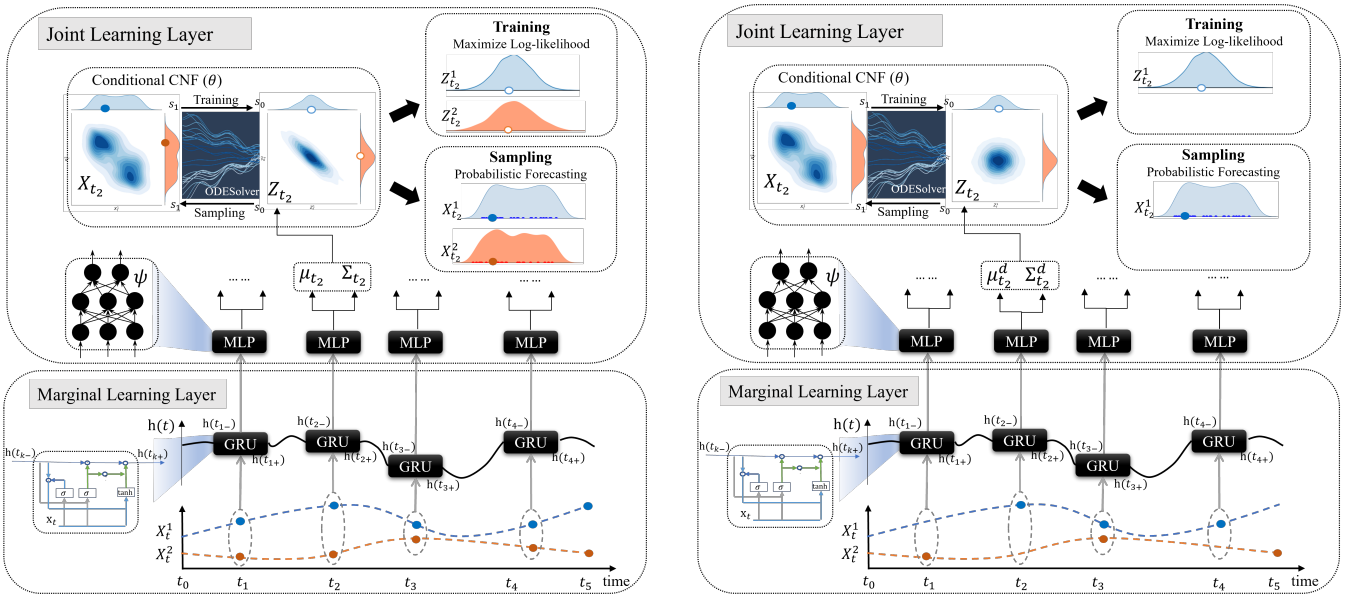
The vanilla GRU has the following updating formulas:

$$\mathrm{h}_t = (1 - \mathrm{z}_t) \odot \tilde{\mathrm{h}}_t + \mathrm{z}_t \odot \mathrm{h}_{t-1}, \tag{4}$$

where z_t, $\tilde{\mathrm{h}}_t$, $\mathrm{h}_t \in \mathbb{R}^H$ are vectors denoting the update gates, the candidate update gates, and the hidden states; the operator $\odot$ denotes the element-wise multiplication.

Subtracting h_{t-1} on both sides of (4) leads to

$$\mathrm{h}_t - \mathrm{h}_{t-1} = (1 - \mathrm{z}_t) \odot (\tilde{\mathrm{h}}_t - \mathrm{h}_{t-1}).$$



(a) Synchronous case (b) Asynchronous case

Fig. 3. The framework of RFNs for (a) Syn-MTS and (b) Asyn-MTS. In both cases, there are two component variables X_t^1, X_t^2. The solid points in different colors indicate they are observations of different variables. In the marginal learning layer, the hidden states will be updated only when at least one variable has an observation, e.g., from $\mathrm{h}(t_{1-})$ to $\mathrm{h}(t_{1+})$. In the joint learning layer, the base distribution parameters at each time are μ_t and Σ_t (for the Syn-MTS case) or μ_t^d and Σ_t^d (for the Asyn-MTS case), which is learned from hidden state $\mathrm{h}(t_{1-})$. The conditional CNF transforms the data points following the unknown distribution to the base distribution, for which likelihoods are easy to compute.

Then, by taking the limit as the time difference between $t-1$ and t tends to zero, one arrives at the continuous-time dynamics of the hidden states:

$$\frac{d\mathrm{h}(t)}{dt} = (1 - \mathrm{z}(t)) \odot (\tilde{\mathrm{h}}(t) - \mathrm{h}(t)). \tag{5}$$

We thus specify equations governing the evolution of the hidden states in two regimes, continuous-updating and discrete-updating.

The *continuous-updating* regime corresponds to continuous time intervals from t_{0+} to t_{1-}, t_{1+} to t_{2-}, t_{2+} to t_{3-}, t_{3+} to t_{4-}, and t_{4+} to t_{5-} in Fig. 3 when no variables have observations. Here, t_- denotes the time point right before t and t_+ means the time point immediately after t. In this regime, we assume that the hidden states for all variables evolve according to equation (5) between observation intervals. To be concrete, the details of evolution are given in equation (6).

The *discrete-updating* regime corresponds to the set of discrete time points $t = t_1, t_2, t_3, t_4$ at which at least one variable is observed. In this case, the hidden state will be updated according to the observations x_{t_k} via vanilla GRU cell, i.e., equation (7).

Continuous-updating:

$$\forall t \in [0, T] \backslash [t_1, \cdots, t_K],$$
$$\frac{d\mathrm{h}(t)}{dt} = (1 - \mathrm{z}^c(t)) \odot (\tilde{\mathrm{h}}^c(t) - \mathrm{h}(t)), \quad s.t. \tag{6}$$
$$\begin{cases} \mathrm{r}^c(t) = \sigma(\mathbf{w}_r^c \mathrm{x}(t) + \mathbf{u}_r^c \mathrm{h}(t) + \mathrm{b}_r^c), \\ \mathrm{z}^c(t) = \sigma(\mathbf{w}_z^c \mathrm{x}(t) + \mathbf{u}_z^c \mathrm{h}(t) + \mathrm{b}_z^c), \\ \tilde{\mathrm{h}}^c(t) = \tanh(\mathbf{w}_h^c \mathrm{x}(t) + \mathbf{u}_h^c (\mathrm{r}^c(t) \odot \mathrm{h}(t)) + \mathrm{b}_h^c). \end{cases}$$

Discrete-updating:

$$\forall t \in [t_1, \cdots, t_K],$$
$$\mathrm{h}(t_+) = (1 - \mathrm{z}^u(t_-)) \odot \tilde{\mathrm{h}}^u(t_-) + \mathrm{z}^u(t_-) \odot \mathrm{h}(t_-) \quad s.t. \tag{7}$$
$$\begin{cases} \mathrm{r}^u(t_-) = \sigma\left(\mathbf{w}_r^u \mathrm{x}_t + \mathbf{u}_r^u \mathrm{h}(t_-) + \mathrm{b}_r^u\right), \\ \mathrm{z}^u(t_-) = \sigma\left(\mathbf{w}_z^u \mathrm{x}_t + \mathbf{u}_z^u \mathrm{h}(t_-) + \mathrm{b}_z^u\right), \\ \tilde{\mathrm{h}}^u(t_-) = \tanh\left(\mathbf{w}_h^u \mathrm{x}_t + \mathbf{u}_h^u (\mathrm{r}^u(t_-) \odot \mathrm{h}(t_-)) + \mathrm{b}_h^u\right). \end{cases}$$

The architecture of the marginal learning layer can be widely adaptable and can be substituted with other sequential models. These alternative formulations are detailed in Appendix B.

B. Joint Synchronous Learning Layer

Once the marginal learning layer is set up, we seek to maximize the log-likelihood of all observations,

$$\max_{\Phi} \log p(\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_K}; \Phi),$$

where p is the joint density to be estimated from the observed data at all observation times $[t_1, \cdots, t_K]$ and $\Phi = \{\phi_1, \cdots, \phi_K\}$ is the parameter set.

1) Factorizing Log-likelihood Objective: If the multivariate sequence is synchronous, observing one variable implies the observation of all other variables, despite potentially unevenly-spaced time intervals. The synchronous property ensures alignments, enabling the use of the chain rule of probability. Thus, we can decompose $p(\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_K}; \Phi)$ as

$$\begin{aligned} & p(\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_K}; \Phi) \\ =& p(\mathrm{x}_{t_1}; \phi_1) p(\mathrm{x}_{t_2} | \mathrm{x}_{t_1}; \phi_2) \cdots p(\mathrm{x}_{t_K} | \mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_{K-1}}; \phi_K) \\ =& p(\mathrm{x}_{t_1}; \phi_1) \prod_{k=2}^{K} p(\mathrm{x}_{t_k} | \mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_{k-1}}; \phi_k). \end{aligned} \tag{8}$$

Such decomposition can be used to factorize the maximization of the joint log-likelihood at all observation times into optimizing conditional log-likelihoods individually:

$$\begin{aligned}&\max_{\Phi} \log p(\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_K}; \Phi) \\ &= \max_{\phi_1} \log p(\mathrm{x}_{t_1}; \phi_1) + \sum_{k=2}^{K} \max_{\phi_k} \log p(\mathrm{x}_{t_k} | \mathrm{x}_{t_{1:k-1}}; \phi_k).\end{aligned}$$

Note that the amount of exogenous information in the hidden state up to t_{k-1} will not increase until the next observation arrives at t_k. Therefore, conditioning on the sample path $\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_{k-1}}$ is equivalent to conditioning on the hidden state at time t_{k-} under mild assumptions. As a result, we have

$$\max_{\Phi} \log p(\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_K}; \Phi) = \sum_{k=1}^{K} \max_{\phi_k} \log p(\mathrm{x}_{t_k} | \mathrm{h}_{t_{k-}}; \phi_k). \tag{9}$$

2) Conditional Flow Representation: Optimizing (9) requires a conditional representation of the data distribution, wherein the log-densities can be tailored to suit one's specific conditional choices. For this purpose, we develop the following conditional formulation of the continuous normalizing flow. Fig. 4 displays the framework of conditional CNF.

Given three random variables X, Y, and Z, our objective is constructing conditional CNF, i.e., expressing conditional log-density $\log p(\mathrm{x}|\mathrm{y})$ in terms of $\log p(\mathrm{z}|\mathrm{y})$ when there is a non-parametric map f that maps z to x under the control of y. Lemma IV.1 is useful to achieve our goal.

Lemma IV.1. *Let $\tilde{\mathrm{z}}(s) = [\mathrm{z}(s), \mathrm{y}(s)]^\top$ be a finite continuous random variable, and the probability density function of $\tilde{\mathrm{z}}(s)$ is $p(\tilde{\mathrm{z}}(s)) = p(\mathrm{z}(s), \mathrm{y}(s))$ which depends on flow time s, where $s_0 \leq s \leq s_1$. Given the governing dynamics of $\tilde{\mathrm{z}}(s)$ as*

$$\frac{\partial \tilde{\mathrm{z}}(s)}{\partial s} = \begin{bmatrix} \frac{\partial \mathrm{z}(s)}{\partial s} \\ \frac{\partial \mathrm{y}(s)}{\partial s} \end{bmatrix} = \begin{bmatrix} \mathrm{f}(\mathrm{z}(s), s, \mathrm{y}(s); \theta) \\ 0 \end{bmatrix},$$

where $s_0 \leq s \leq s_1$, $\mathrm{z}(s_1) = \mathrm{x}$, $\mathrm{y}(s_1) = \mathrm{y}$, and f is Lipschitz continuous in z and continuous in s for any y. We have

$$\begin{aligned}&\log p(\mathrm{x}, \mathrm{y}) \\ &= \log p(\mathrm{z}(s_0), \mathrm{y}) + \int_{s_1}^{s_0} \mathrm{Tr}[\partial_{\mathrm{z}(s)} \mathrm{f}(\mathrm{z}(s), s, \mathrm{y}; \theta)] ds.\end{aligned} \tag{10}$$

Proof. See the Appendix D. □

Subtracting $\log p(\mathrm{y})$ on both sides of equation (10) in Lemma IV.1 and using the fact that $\log p(\mathrm{x}, \mathrm{y}) - \log p(\mathrm{y}) = \log \frac{p(\mathrm{x},\mathrm{y})}{p(\mathrm{y})} = \log p(\mathrm{x}|\mathrm{y})$, we obtain a result of conditional CNF that is summarized in Proposition IV.1.

Proposition IV.1. *Let the assumptions in Lemma IV.1 hold. The conditional log-density $p(\mathrm{x}|\mathrm{y})$ is given by:*

$$\begin{aligned}&\log p(\mathrm{x}|\mathrm{y}) \\ &= \log p(\mathrm{z}(s_0)|\mathrm{y}) + \int_{s_1}^{s_0} \mathrm{Tr}[\partial_{\mathrm{z}(s)} \mathrm{f}(\mathrm{z}(s), s, \mathrm{y}; \theta)] ds.\end{aligned} \tag{11}$$

By extending the unconditional CNF formulation (see Section II-B) to the conditional formulation, equation (11) allows the data distribution to be both non-parametric via the flow map f and time-varying, e.g., by choosing the conditional information y properly.

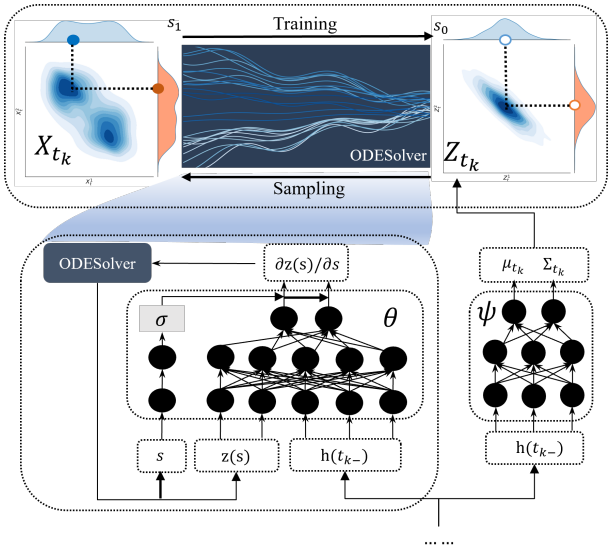


Fig. 4. The framework of conditional CNF.

3) Time-varying Specification: We now discuss how to obtain $p(\mathrm{x}_{t_k}|\mathrm{h}_{t_{k-}}; \phi_k)$ in equation (9) via conditional CNF. First, the conditional information y in equation (11) is the hidden states $\mathrm{h}_{t_{k-}}$. We assume that there is a bijective map between x_{t_k} and z_{t_k} such that $\mathrm{z}_{t_k}|\mathrm{h}_{t_{k-}}$ is Gaussian distributed. Further, we shall let the hidden states determine the parameter values of the Gaussian base distribution. Because hidden states contain information on historical data and vary at different observation times, this choice of conditioning leads to path-dependent parameters of the base distribution.

Specifically, we assume that the mean vector μ_{t_k} and covariance matrix Σ_{t_k} of the Gaussian base are functions (learned by standard MLP, denoted by g) of the hidden states $\mathrm{h}_{t_{k-}}$ at time-t_{k-} learned from the marginal learning layer, i.e.,

$$\begin{aligned}&\mathrm{Z}_{t_k}|\mathrm{h}_{t_{k-}} \sim \mathcal{N}(\mu_{t_k}, \Sigma_{t_k}), \quad \text{where} \\ &\{\mu_{t_k}, \Sigma_{t_k}\} = \mathrm{g}(\mathrm{h}_{t_{k-}}; \psi).\end{aligned} \tag{12}$$

This is in contrast with the unconditional flow models where the base distribution parameters are constants as it is a standard Normal $\mathcal{N}(0, \mathbb{I}_D)$.

It is worth noting that previous studies [16, 19, 20] also use hidden states to learn Gaussian parameters. However, those Gaussian parameters are associated with the data distribution. Our representation shifts the Gaussian assumption from the data distribution to the base distribution, and it is the base distribution parameters that depend on hidden states. This distinction is crucial because our representation enables the data distribution to be non-parametric, non-Gaussian, and time-varying simultaneously.

Meanwhile, we use the following gated mechanism to incorporate the hidden state dependence into the flow map. Mathematically, we have

$$\begin{aligned}&\mathrm{f}(\mathrm{z}(s), s, \mathrm{h}_{t_{k-}}; \theta) \\ &= (\mathbf{w}_z \mathrm{z}(s) + \mathbf{w}_h \mathrm{h}_{t_{k-}} + \mathrm{b}_z) \sigma(w_s s + b_s).\end{aligned} \tag{13}$$

Here, σ is the sigmoid activation function. θ is the set of trainable parameters which includes $\mathbf{w}_h \in \mathbb{R}^{D \times H}$, $\mathrm{b}_z \in \mathbb{R}^D$, and $\{w_s, b_s\} \in \mathbb{R}$. The resulting flow dynamics is

$$\frac{\partial \mathrm{z}(s)}{\partial s} = \mathrm{f}(\mathrm{z}(s), s, \mathrm{h}_{t_{k-}}; \theta), s \in [s_0, s_1],$$
$$\text{where} \quad \mathrm{z}(s)|_{s=s_0} = \mathrm{z}_{t_k},\ \mathrm{z}(s)|_{s=s_1} = \mathrm{x}_{t_k}.$$

The initial value of the flow $\mathrm{z}(s)|_{s=s_0}$ is set as the Gaussian base sample z_{t_k}, sampled from $\mathrm{Z}_{t_k}|\mathrm{h}_{t_{k-}}$ per (12), and the terminal flow value $\mathrm{z}(s)|_{s=s_1}$ is set to equal the time-t_k observed data x_{t_k}. Its solution maps samples from base distribution to data distribution concurrently, given as

$$\mathrm{x}_{t_k} = \mathrm{z}_{t_k} + \int_{s_0}^{s_1} \mathrm{f}(\mathrm{z}(s), s, \mathrm{h}_{t_{k-}}; \theta) ds. \tag{14}$$

We summarize the time-varying specification of the conditional CNF representation for Syn-MTS data as follows. For $t_k \in [t_1, \cdots, t_K]$, we have

$$\begin{bmatrix} \mathrm{x}_{t_k} \\ \log p(\mathrm{x}_{t_k}|\mathrm{h}_{t_{k-}}; \phi_{t_k}) \end{bmatrix} = \begin{bmatrix} \mathrm{z}_{t_k} \\ \log p(\mathrm{z}_{t_k}|\mathrm{h}_{t_{k-}}; \mu_{t_k}, \Sigma_{t_k}) \end{bmatrix} + \int_{s_0}^{s_1} \begin{bmatrix} \mathrm{f}\big(\mathrm{z}(s), s, \mathrm{h}_{t_{k-}}; \theta\big) \\ -\mathrm{Tr}[\partial_{\mathrm{z}(s)}\mathrm{f}] \end{bmatrix} ds. \tag{15}$$

It has two essential components, the time-varying base distribution driven by hidden states where ψ in (12) is the parameter to be trained, and the time-varying flow map indexed by observation arrival times with θ in (13) being the set of trainable parameters.

4) Training and Sampling: To train the NFR model, we maximize the conditional log-likelihoods (9) across all sample instances at all observation times. We set the dataset level log-likelihood $\mathcal{L}_{\text{Syn-MTS}}$ as the sample average of $(\mathcal{L}^i_{\text{Syn-MTS}})_{i=1}^N$, where each $\mathcal{L}^i_{\text{Syn-MTS}}$ is weighted by the number of observation times K_i. To be concrete, we have

$$\begin{aligned} &\mathcal{L}_{\text{Syn-MTS}} \\ =&\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}^i_{\text{Syn-MTS}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K_i}\sum_{k=1}^{K_i}\log p(\mathrm{x}_{i,t^i_k}|\mathrm{h}_{i,t^i_{k-}}; \phi_{i,t^i_k}) \\ =&\frac{1}{N}\sum_{i=1}^{N}\frac{1}{K_i}\sum_{k=1}^{K_i}\begin{pmatrix} \log p(\mathrm{z}_{i,t^i_k}|\mathrm{h}_{i,t^i_{k-}}; \mu_{i,t^i_k}, \Sigma_{i,t^i_k}) + \\ \int_{s_1}^{s_0} \mathrm{Tr}\left[\partial_{\mathrm{z}(s)}\mathrm{f}(\mathrm{z}(s), s, \mathrm{h}_{i,t^i_{k-}}; \theta)\right] ds \end{pmatrix}. \end{aligned} \tag{16}$$

Once the model is trained, we can forecast the joint data distribution at time point t_k given the observations $\{\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_{k-1}}\}$ of the i^{th} instance. The confidence interval of predictions can be obtained by sampling from the learned joint distribution.

For sampling, we first obtain the hidden states $\mathrm{h}_{t_{k-}}$ for the i^{th} instance based on the observations $\{\mathrm{x}_{t_1}, \cdots, \mathrm{x}_{t_{k-1}}\}$ per (6) and (7), and use $\mathrm{h}_{t_{k-}}$ to predict the base distribution parameters $\{\mu_{t_k}, \Sigma_{t_k}\}$ per (12). We then sample a given number of points z_{t_k} from the base distribution with the predicted parameters. Finally, we transform these base sample points using the conditional CNF map (14) to obtain concurrent samples that follow the data distribution.

C. Joint Asynchronous Learning Layer

If the multivariate sequence is asynchronous, it implies the possibility that not all entries of the vector x_{t_k} are observable at any given time $t_k \in \mathrm{t}$ (as defined in Definition 2).

1) Masked Independent Log-likelihoods Objective: In the scenario of Syn-MTS, it is observed that all variables simultaneously record observations $\mathrm{x}_{t_k} \in \mathbb{R}^D$ at a specific observational time point t_k. Under these circumstances, one can convert it into a sample $\mathrm{z}_{t_k} \in \mathbb{R}^D$ following multivariate Gaussian distribution conditioned on the hidden state $\mathrm{h}_{t_{k-}}$ via a conditional CNF. The mean and covariance of this multivariate Gaussian distribution are represented as μ_{t_k} and Σ_{t_k}, respectively. Consequently, the log-likelihood of x_{t_k} can be computed as

$$\begin{aligned} &\log p(\mathrm{x}_{t_k}|\mathrm{h}_{t_{k-}}; \phi_{t_k}) \\ =&\log p(x^1_{t_k}, \cdots, x^D_{t_k}|\mathrm{h}_{t_{k-}}; \phi_{t_k}) \\ =&\log p(z^1_{t_k}, \cdots, z^D_{t_k}|\mathrm{h}_{t_{k-}}; \mu_{t_k}, \Sigma_{t_k}) + \int_{s_0}^{s_1} -\mathrm{Tr}[\partial_{z(s)}\mathrm{f}] ds. \end{aligned}$$

In the Asyn-MTS cases, where some variables do not have values at given times, the approach used in the synchronous case cannot be applied directly. To resolve this additional complication, we restrict each dimension of conditional multivariate Gaussian distribution to be independent. This allows for the decomposition of the joint log-likelihood $\log p(z^1_{t_k}, \cdots, z^D_{t_k}|\mathrm{h}_{t_{k-}}; \mu_{t_k}, \Sigma_{t_k})$ into individual components, as follows

$$p(z^1_{t_k}, \cdots, z^D_{t_k}|\mathrm{h}_{t_{k-}}; \mu_{t_k}, \Sigma_{t_k}) = \prod_{d=1}^{D} p(z^d_{t_k}|\mathrm{h}_{t_{k-}}; \mu^d_{t_k}, \Sigma^d_{t_k}).$$

Thus, we have

$$\begin{aligned} &\log p(x^1_{t_k}, \cdots, x^D_{t_k}|\mathrm{h}_{t_{k-}}; \phi_{t_k}) \\ =&\log \prod_{d=1}^{D} p(z^d_{t_k}|\mathrm{h}_{t_{k-}}; \mu^d_{t_k}, \Sigma^d_{t_k}) + \int_{s_0}^{s_1} -\mathrm{Tr}[\partial_{z(s)}\mathrm{f}] ds \\ =&\sum_{d=1}^{D} \log p(z^d_{t_k}|\mathrm{h}_{t_{k-}}; \mu^d_{t_k}, \Sigma^d_{t_k}) + \int_{s_0}^{s_1} -\mathrm{Tr}[\partial_{z(s)}\mathrm{f}] ds. \end{aligned} \tag{17}$$

Note that $(\mu^d_{t_k}, \Sigma^d_{t_k}) \in \mathbb{R} \times \mathbb{R}$ are learned from MLPs. For each d, $\mu^d_{t_k}$ and $\Sigma^d_{t_k}$ represent the mean and variance of the variable $Z^d_{t_k}$ conditioned on the hidden state $\mathrm{h}_{t_{k-}}$, which is assumed to follow a Gaussian distribution.

Nevertheless, it is noteworthy that at a specific observation time t_k, not all variables $x^d_{t_k}$ are necessarily observed. For any unobserved variable $x^d_{t_k}$, its corresponding log-density should be omitted from (17). To address this scenario in practice, we employ a masking mechanism denoted as $m^d_{t_k}$ to mitigate the impact of the unobserved quantity $\log p(z^d_{t_k}|\mathrm{h}_{t_{k-}}; \mu^d_{t_k}, \Sigma^d_{t_k})$. As a result, equation (17) is transformed into the form:

$$\begin{aligned} &\log p(z^d_{t_k}|\mathrm{h}_{t_{k-}}; \mathrm{m}_{t_k}, \phi_{t_k}) \\ =&\sum_{d=1}^{D} m^d_{t_k} \log p(z^d_{t_k}|\mathrm{h}_{t_{k-}}; \mu^d_{t_k}, \Sigma^d_{t_k}) + \int_{s_0}^{s_1} -\mathrm{Tr}[\partial_{z(s)}\mathrm{f}] ds. \end{aligned} \tag{18}$$

This decomposition along the component dimension facilitates the optimization process by focusing solely on the conditional distribution of variables with observations at time t_k, rather than optimizing across all D variables. Essentially, any component lacking an observation at t_k is effectively masked out,

thereby ensuring its exclusion from influencing the overall loss function.

In addition to the decomposition of the base Gaussian distribution, it is equally crucial to address how unobserved components are handled during the transformation process $\mathrm{x}_{t_k} = \mathrm{f}(\mathrm{z}_{t_k})$. Allowing transformations of missing values in parallel with observed data can significantly skew the learning outcomes. This is attributed to the fact that the transformation applied to these unobserved values inadvertently influences the cumulative transformation loss, specifically $\int_{s_0}^{s_1} -\operatorname{Tr}[\partial_{z(s)}\mathrm{f}]ds$. To mitigate the problem, we allow the transformation process for those observed components, and keep the process unchanged for those unobserved components at each $t_k \in \mathrm{t}$. Thus, the evolution equation of $z^d(s)$ for each d along the flow time can be described by the following ODE:

$$\frac{\partial z^d(s)}{\partial s} = \mathrm{f}(z^d(s), s, \mathrm{h}_{t_{k-}}; m^d_{t_k}, \theta)\ , \quad s \in [s_0, s_1], \tag{19}$$

where

$$\mathrm{f}(z^d(s), s, \mathrm{h}_{t_{k-}}; m^d_{t_k}, \theta) = \begin{cases} \mathrm{f}(z^d(s), s, \mathrm{h}_{t_{k-}}; \theta) & \text{if } m^d_{t_k} = 1, \\ 0 & \text{if } m^d_{t_k} = 0. \end{cases}$$

We summarize the conditional CNF representation for Asyn-MTS data as

$$\begin{aligned} &\begin{bmatrix} \mathrm{x}_{t_k} \\ \log p(\mathrm{x}_{t_k}|\mathrm{h}_{t_{k-}}; \mathrm{m}_{t_k}, \phi_{t_k}) \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{z}_{t_k} \\ \sum_{d=1}^{D} m^d_{t_k} \log p(z^d_{t_k}|\mathrm{h}_{t_{k-}}; \Sigma^d_{t_k}, \mu^d_{t_k}) \end{bmatrix} + \int_{s_0}^{s_1} \begin{bmatrix} \mathrm{f}(\mathrm{z}(s), s, \mathrm{h}_{t_{k-}}, \mathrm{m}_{t_k}; \theta) \\ -\operatorname{Tr}\left[\partial_{z^d(s)}\mathrm{f}\right] \end{bmatrix} ds. \end{aligned} \tag{20}$$

2) Training and Sampling: The target in the asynchronous case also maximizes the log-likelihood of observations. However, different from the objective function (15) of the Syn-MTS model, the unobserved components are masked. Mathematically, we have

$$\begin{aligned} &\mathcal{L}_{\text{Asyn-MTS}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}^i_{\text{Asyn-MTS}}, \quad \text{where} \\ &\mathcal{L}^i_{\text{Asyn-MTS}} \\ &= \sum_{\substack{1\le k\le K_i \\ 1\le d\le D}} \frac{1}{K_i D} \left(\begin{matrix} m^d_{i,t^i_k} \log p(z^d_{i,t^i_k}|\mathrm{h}_{i,t^i_{k-}}; \mu^d_{i,t^i_k}, \Sigma^d_{i,t^i_k}) + \\ \int_{s_1}^{s_0} \operatorname{Tr}[\partial_{z^d(s)}\mathrm{f}(z^d(s), s, \mathrm{h}_{t_{k-}}; m^d_{t_k}, \theta)] ds \end{matrix} \right). \end{aligned} \tag{21}$$

The sampling process at time t_k of the Asyn-MTS model is similar to the Syn-MTS model. We first predict the parameters of base distribution of variables $X^1_{t_k}, \cdots, X^D_{t_k}$ conditional on the hidden state $\mathrm{h}_{t_{k-}}$ from marginal learning layer, i.e., $\{\mu^1_{t_k}, \Sigma^1_{t_k}\}, \cdots, \{\mu^D_{t_k}, \Sigma^D_{t_k}\}$. Then, we sample a given number of points $z^1_{t_k}, \cdots, z^D_{t_k}$ from the base distributions with the predicted parameters, and we transform these points using the conditional CNF model.

The algorithms for the training and sampling procedures of the Syn-MTS model and the Asyn-MTS model are provided in Appendix G.

V. Experiments

We conduct one simulation experiment to verify the efficacy of the model specification, followed by three real-world datasets to evaluate the overall performance of the RFNs. In all studies and experiments, we use 70%/15%/15% of the sample instances for training/validation/testing.

A. Baseline Models

We establish four baseline models specifically to facilitate the construction of the marginal learning layer, namely GRUODE [16], GRU-D [10], ODERNN [14], and ODELSTM [17]. These baselines are all designed to model multivariate irregularly sampled time series. Following [16], We adapt them by assuming that each variable follows a multivariate normal distribution, and thus their joint learning layer is designed to learn the parameters of these normal distributions. Specifically, the parameters of the normal distribution, μ_{t_k} and σ_{t_k} are predicted based on the hidden state $\mathrm{h}(t_k)$ via a feed-forward neural network directly.

To compare the vanilla form of a baseline model, e.g., GRU-D, with its RFN counterpart, e.g., termed as RFN-GRU-D, we use the corresponding RFN specifications laid out in Section IV-B and IV-C for the multivariate learning. For a fair and robust comparison, we employ the same marginal learning layer across all baseline models in RFN models. The only difference between the baselines and our RFN model is the replacement of the joint learning layer—from the multivariate normal distribution assumption in the baseline models to the conditional CNF in RFN. By doing so, we can isolate and demonstrate the specific performance improvement brought by our proposed joint learning layer.

The alternative construction of the marginal learning layer except GRUODE is listed in Appendix B. Hyperparameters, such as learning rate and batch size, are tuned using a random search for all experiments. The 'dopri5' method is employed as the numerical solver for the ODE-based models.

B. Evaluation Metric

We use the Continuous Ranked Probability Score (CRPS) [33] as the evaluation metric to measure the proximity between the learned distribution and the empirical distribution of data. The CRPS metric is the suitable scoring function for the evaluation since it attains the minimum when the learned distribution of data aligns with its empirical distribution. The CRPS is defined as

$$\mathrm{CRPS}(\hat{F}, x) = \int_{\mathbb{R}} (\hat{F}(z) - \mathbb{I}\{x \le z\})^2 \, \mathrm{d}z,$$

where $\hat{F}$ is the estimated CDF of random variable X and x is the realized observations of X. In our dataset, the observations are irregularly sampled, and some values are missing, and thus we only compute CRPS when the variables can be observed.

Since CRPS targets to evaluate the estimated distribution of the univariate random variable, [20] suggests using $\mathrm{CRPS}_{\text{sum}}$ as a new proper multivariate-scoring rule to evaluate the estimated distribution of multivariate random variables. Specifically, we have

$$\mathrm{CRPS}_{\text{sum}} = \mathbb{E}_t \left[\mathrm{CRPS}\left(\hat{F}_{\text{sum}}(t), \sum_{d=1}^{D} x^d_t \right) \right],$$

where $\hat{F}_{\text{sum}}$ is computed by adding the estimated CDF of all variables together. In practice, the empirical CDF is always used to represent $\hat{F}$ and $\hat{F}_{\text{sum}}$. In our experiments, we take 100 samples from trained models to estimate the empirical CDF. For Asyn-MTS datasets, some variables are not observable at an observation time; we thus only add the estimated CDF of observed variables as $\hat{F}_{\text{sum}}$.

Apart from CRPS and CRPS_{sum}, Kuleshov et al. [34] propose a more interpretive metric, termed confidence score (CS), to evaluate the calibration performance of the predicted distribution in the time series forecasting. The key idea behind this metric is to measure how the true frequency of observations across all time points matches the quantiles of the predicted distribution. For each variable d at each time point t, the model predicts a distribution $F_t^d(\cdot)$. From this, we can compute the q-th quantile of the distribution, denoted by $Q_t^d(q) = \inf\{x_t^d \in \mathbb{R} \mid F_t^d(q) \geq x_t^d\}$. For a well-calibrated model, the proportion of actual observations x_t^d smaller than the predicted $Q_t^d(q)$ should closely approximate q.

Specifically, it first choose m quantile levels $0 < p_1 < \cdots < p_m < 1$, then for each threshold p_j, they compute the empirical frequency

$$\hat{p}_j^d = \frac{|\{x_t^d : F_t^d(x_t^d) \leq p_j, t = 1, \cdots, T|\}|}{T}.$$

Then, the confidence score is defined to describe the quality of forecast calibration $\text{CS} = \frac{1}{mD}\sum_{j=1}^{m}\sum_{d=1}^{D}(p_j - \hat{p}_j^d)^2$.

C. Simulation Experiment

(a) Data Generating Process. We set the Data Generation Process (DGP) $\text{X}_\text{t} = [X_t^1, \cdots, X_t^5]$ as the following continuous-time correlated Geometric Brownian Motion (GBM) [35]

$$d\text{X}_t = \mu \odot \text{X}_t dt + \text{diag}(\sigma \odot \text{X}_t) d\text{W}_t,\ t \geq 0, \qquad (22)$$

where $\mu = [\mu^1, \cdots, \mu^5]^\top$ and $\sigma = [\sigma^1, \cdots, \sigma^5]^\top$ are the drift term and volatility term of variables, and $\text{W}_t = [W_t^1, \cdots, W_t^5]^\top$ is the multivariate Brownian Motion process such that $\langle dW_t^i, dW_t^j \rangle = \rho_{ij}(t)dt$ where

$$[\rho_{ij}(t)]_{1\leq i,j\leq 5} := \sin\left(\frac{\pi}{2}t\right)\begin{bmatrix} 1 & \rho_1 & 0 & 0 & 0 \\ \rho_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \rho_2 & \rho_2 \\ 0 & 0 & \rho_2 & 1 & \rho_2 \\ 0 & 0 & \rho_2 & \rho_2 & 1 \end{bmatrix}.$$

We devise such a correlated GBM as the DGP for several reasons. Firstly, the GBM is a fundamental model in physics and asset prices in financial markets [36]. Secondly, the correlated GBM admits log-normal distribution for each variable, distinguishing it from the normal distribution. Thirdly, the correlation among the five variables is not set to static but time-varying, gradually increasing as time progresses from 0 to 1 due to the sinusoidal function $\sin(\frac{\pi}{2}t)$. This guarantees that the joint distribution of the five variables is constantly changing. Fig. 5(a) showcases three representative sample paths from the dataset. Additionally, Fig. 5(b) illustrates the sample correlation matrix at observed time points 0.3, 0.6, and 0.9, validating the dynamic and gradual increase in correlation.

(b) Simulation Settings. We first simulate 1,000 sample paths using (22) with uniform time intervals. The Brownian parameters, i.e., the drift terms and the diffusion terms, of the 1,000 sample paths are set as follows: $\mu^1 = \mu^2 \sim \text{Unif}(-0.2, -0.05), \mu^3 = \mu^4 = \mu^5 \sim \text{Unif}(0.05, 0.2), \sigma^1 = \sigma^2 \sim \text{Unif}(0.15, 0.3), \sigma^3 = \sigma^4 = \sigma^5 \sim \text{Unif}(0.15, 0.3)$, where $\text{Unif}(\cdot, \cdot)$ stands for a uniform distribution.

We create the Syn-MTS and Asyn-MTS datasets through random sampling from the simulated dataset with complete observation. In the Syn-MTS dataset, we randomly choose half of the time points and sample the corresponding observations X_t at the chosen time points. To generate the Asyn-MTS dataset, we randomly eliminate half of the observations for each variable. The elimination is independent between any two variables. Therefore, some variables have observations at an observed time point, and some variables do not.

(c) Data and Model Validation. The experiment results in Table I demonstrate that RFNs outperform the corresponding baselines with smaller CRPS, CRPS_{sum}, and CS. This indicates the superior ability of the RFNs to capture the joint distribution of multivariate irregular time series at each observed time point and recover its dynamics. Fig. 6(a) and 6(b) illustrate the correlation matrices sampled from the estimated Syn-MTS and Asyn-MTS models at time points 0.3, 0.6, and 0.9, closely resembling the ground truth in Fig. 5(b). The correlation coefficients from the recovered matrices gradually transition from weak to strong as time progresses, aligning with the ground truth in Fig. 5(b). This dynamic capture of the joint distribution validates the effectiveness of the RFN specification. Lastly, we employ the RFNs for interval estimation of correlated GBM processes. Fig. 7(a) and Fig. 7(b) display prediction intervals for Syn-MTS and Asyn-MTS.

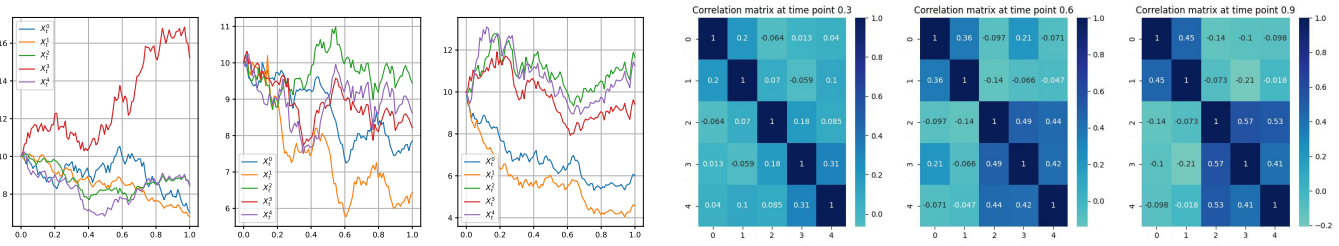
D. Dataset of Physical Activities (MuJoCo)

In applications such as robotics, multivariate time series are commonly used to capture the position, speed, and trajectory of objects, exhibiting a high degree of correlation among variables. However, irregular and asynchronous frequency of measurements frequently arises due to limitations in measurement devices. This experiment tests the performance of using the RFN specification to predict object positions against baseline models in their vanilla forms.

The MuJoCo physics dataset is introduced in [14] to verify that the ODE-based models can learn an approximation of Newtonian physics. The dataset is created by the "Hopper" model from the Deepmind Control Suite [37]. It contains 14 variables in total, where the first 7 variables and the last 7 variables control the position and the velocities.

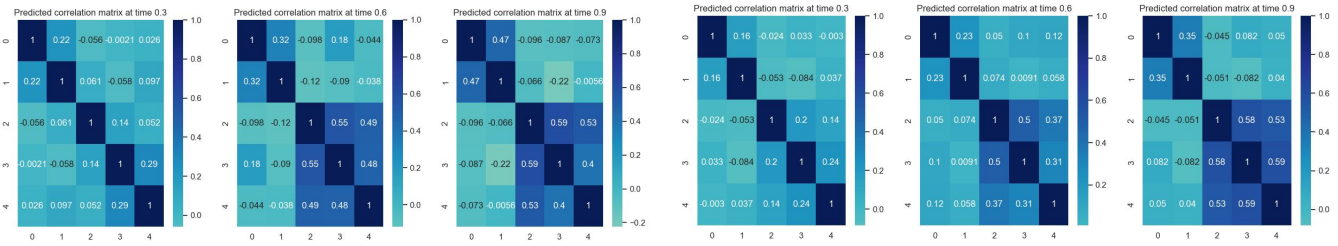
By following the generation and preprocessing of [14], we randomly sample the initial position and velocities of "Hopper" such that the hopper rotates in the air and falls on the ground. We generate 5,000 instances with 150 observed time points for each instance, and randomly sample 50% values to generate the Syn-MTS and Asyn-MTS datasets.

Table II shows that the RFNs perform robustly better than the baselines in their vanilla form. As a validation, we use



(a) (b)

Fig. 5. (a) Three representative sample paths. (b) The sample correlation matrix at observed time points 0.3, 0.6, 0.9.



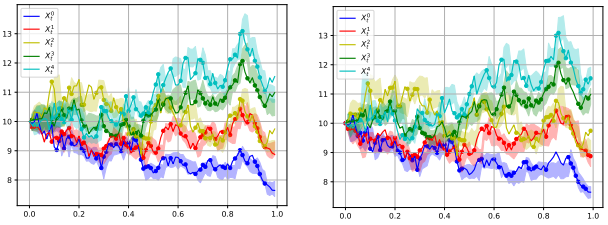
(a) (b)

Fig. 6. The sample correlation matrix recovered from the learned Syn-MTS model (a) and the learned Asyn-MTS model (b) at 0.3, 0.6, and 0.9.

TABLE I

GEOMETRIC BROWNIAN MOTIONS: BASELINE MODELS IN VANILLA FORM (2,3,4 COLUMNS) VS. USING RFN SPECIFICATION (5,6,7 COLUMNS). NUMBERS UNDERNEATH CRPS, CRPS_{SUM} , CS ARE DISPLAYED IN THE FORM OF MEAN $\pm$ STD, AVERAGED OVER FIVE EXPERIMENTS.

model	CRPS	CRPS_{sum}	CS	model	CRPS	CRPS_{sum}	CS
			Syn-MTS				
GRUODE	0.2051 $\pm$ 0.0182	0.6785 $\pm$ 0.0120	0.0010 $\pm$ 0.0003	RFN-GRUODE	0.1870 $\pm$ 0.0020	0.5869 $\pm$ 0.0088	0.0007 $\pm$ 0.0002
ODELSTM	0.2049 $\pm$ 0.0190	0.6697 $\pm$ 0.0120	0.0003 $\pm$ 0.0001	RFN-ODELSTM	0.1826 $\pm$ 0.0007	0.5795 $\pm$ 0.0025	0.0002 $\pm$ 0.0001
ODERNN	0.2070 $\pm$ 0.0045	0.6777 $\pm$ 0.0157	0.0002 $\pm$ 0.0001	RFN-ODERNN	0.1808 $\pm$ 0.0017	0.5744 $\pm$ 0.0078	0.0001 $\pm$ 0.0001
GRU-D	0.1921 $\pm$ 0.0130	0.6372 $\pm$ 0.0077	0.0015 $\pm$ 0.0006	RFN-GRU-D	0.1838 $\pm$ 0.0007	0.5815 $\pm$ 0.0071	0.0015 $\pm$ 0.0003
			Asyn-MTS				
GRUODE	0.2308 $\pm$ 0.0046	0.4565 $\pm$ 0.0125	0.0006 $\pm$ 0.0001	RFN-GRUODE	0.2045 $\pm$ 0.0042	0.3827 $\pm$ 0.0117	0.0003 $\pm$ 0.0002
ODELSTM	0.2261 $\pm$ 0.0061	0.4677 $\pm$ 0.0326	0.0002 $\pm$ 0.0001	RFN-ODELSTM	0.1962 $\pm$ 0.0015	0.3650 $\pm$ 0.0033	0.0001 $\pm$ 0.0001
ODERNN	0.2160 $\pm$ 0.0051	0.4283 $\pm$ 0.0144	0.0004 $\pm$ 0.0001	RFN-ODERNN	0.1918 $\pm$ 0.0048	0.3597 $\pm$ 0.0130	0.0001 $\pm$ 0.0001
GRU-D	0.2220 $\pm$ 0.0057	0.4420 $\pm$ 0.0162	0.0010 $\pm$ 0.0002	RFN-GRU-D	0.1858 $\pm$ 0.0019	0.3405 $\pm$ 0.0041	0.0007 $\pm$ 0.0001



(a) Syn-MTS (b) Asyn-MTS

Fig. 7. Predicted quantile intervals (shaded area) from 20% to 80% of Syn-MTS data (a) and Asyn-MTS data (b) using the RFN-GRUODE specification. Solid dots represent observations. Broader quantile ranges are available in Appendix F.

the trained model to forecast the future joint distribution at $t = 1.40$ based on a given instance's past ten observations (ranging from $t = 0.09$ to $t = 1.17$). Subsequently, we generate 10 samples from the estimated distribution at $t = 1.40$ and compare them with the ground truth at $t = 1.40$. Fig. 9 displays the 10 samples at $t = 1.40$ from both the Asyn-MTS model and the Syn-MTS model. They all resemble the ground truth closely, affirming the RFN's proficiency in predicting the joint distribution.

In Fig. 8(a) and 8(b), we present a comparison based on the CS metric to understand the model's reliability. The x-axis represents the observed quantile level, while the y-axis shows the predicted quantile level. The solid black line represents the theoretical best performance, where the predicted quantile exactly matches the observed quantile, i.e., the x-axis equals the y-axis. The closer a model's performance is to this black line, the better it is calibrated. We plot the results for our model (solid lines) alongside the baseline models (dashed lines). We observe that almost all the dashed lines fall upper and to the left of the solid lines, indicating that our models outperform

TABLE II
PHYSICAL ACTIVITIES OF HUMAN BODY (MUJOCO). (SAME REPORT FORMATS AS IN TABLE 1)

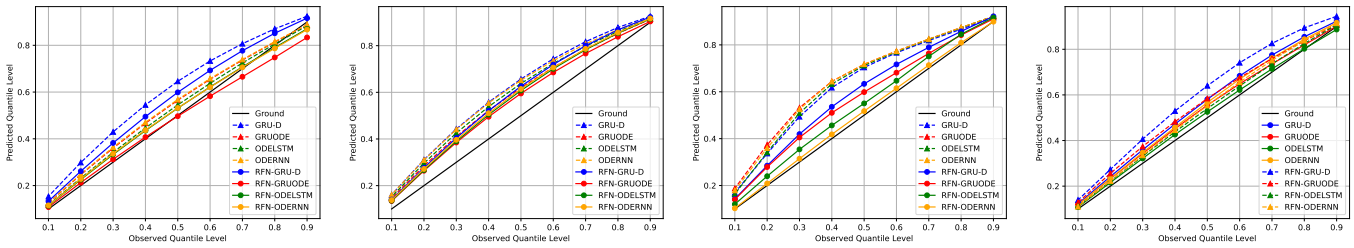
model	CRPS	CRPS_{sum}	CS	model	CRPS	CRPS_{sum}	CS
			Syn-MTS				
GRUODE	0.2198 ± 0.0035	1.0763 ± 0.0352	0.0117 ± 0.0022	RFN-GRUODE	0.1858 ± 0.0037	0.6790 ± 0.0300	0.0097 ± 0.0033
ODELSTM	0.2256 ± 0.0015	1.1402 ± 0.0298	0.0117 ± 0.0026	RFN-ODELSTM	0.1736 ± 0.0017	0.7103 ± 0.0504	0.0060 ± 0.0010
ODERNN	0.2156 ± 0.0042	1.0450 ± 0.0393	0.0143 ± 0.0016	RFN-ODERNN	0.1747 ± 0.0019	0.6467 ± 0.0340	0.0087 ± 0.0025
GRU-D	0.2224 ± 0.0032	1.1118 ± 0.0177	0.0119 ± 0.0014	RFN-GRU-D	0.1905 ± 0.0010	0.6748 ± 0.0093	0.0084 ± 0.0007
			Asyn-MTS				
GRUODE	0.2695 ± 0.0054	1.0661 ± 0.0249	0.0024 ± 0.0001	RFN-GRUODE	0.1970 ± 0.0052	0.7306 ± 0.0185	0.0007 ± 0.0002
ODELSTM	0.2728 ± 0.0057	1.0834 ± 0.0292	0.0013 ± 0.0002	RFN-ODELSTM	0.1733 ± 0.0048	0.6318 ± 0.0155	0.0007 ± 0.0002
ODERNN	0.2696 ± 0.0036	1.0748 ± 0.0194	0.0020 ± 0.0005	RFN-ODERNN	0.1667 ± 0.0020	0.6277 ± 0.0086	0.0012 ± 0.0004
GRU-D	0.2261 ± 0.0035	0.9078 ± 0.0255	0.0116 ± 0.0008	RFN-GRU-D	0.1709 ± 0.0035	0.6409 ± 0.0151	0.0058 ± 0.0009

TABLE III
CLIMATE RECORDS OF WEATHER (USHCN). (SAME REPORT FORMATS AS IN TABLE 1)

model	CRPS	CRPS_{sum}	CS	model	CRPS	CRPS_{sum}	CS
GRUODE	0.3012 ± 0.0082	0.5797 ± 0.0112	0.0269 ± 0.0043	RFN-GRUODE	0.2463 ± 0.0039	0.5183 ± 0.0072	0.0075 ± 0.0052
ODELSTM	0.3113 ± 0.0049	0.5909 ± 0.0075	0.0256 ± 0.0029	RFN-ODELSTM	0.2629 ± 0.0043	0.5510 ± 0.0090	0.0071 ± 0.0037
ODERNN	0.3112 ± 0.0088	0.5923 ± 0.0157	0.0285 ± 0.0011	RFN-ODERNN	0.2626 ± 0.0073	0.5499 ± 0.0146	0.0051 ± 0.0046
GRU-D	0.3016 ± 0.0143	0.5798 ± 0.0186	0.0243 ± 0.0038	RFN-GRU-D	0.2415 ± 0.0016	0.5093 ± 0.0025	0.0094 ± 0.0051

TABLE IV
TRANSACTION RECORDS OF EIGHT BIOTECHNOLOGY STOCKS (NASDAQ). (SAME REPORT FORMATS AS IN TABLE 1)

model	CRPS	CRPS_{sum}	CS	model	CRPS	CRPS_{sum}	CS
GRUODE	0.0163 ± 0.0028	0.0578 ± 0.0219	0.0021 ± 0.0009	RFN-GRUODE	0.0149 ± 0.0008	0.0514 ± 0.0039	0.0007 ± 0.0006
ODELSTM	0.0163 ± 0.0021	0.0529 ± 0.0085	0.0013 ± 0.0010	RFN-ODELSTM	0.0150 ± 0.0009	0.0506 ± 0.0056	0.0004 ± 0.0002
ODERNN	0.0166 ± 0.0009	0.0564 ± 0.0047	0.0031 ± 0.0020	RFN-ODERNN	0.0150 ± 0.0008	0.0521 ± 0.0036	0.0021 ± 0.0009
GRU-D	0.0068 ± 0.0007	0.0270 ± 0.0016	0.0102 ± 0.0042	RFN-GRU-D	0.0064 ± 0.0003	0.0244 ± 0.0005	0.0049 ± 0.0054



(a) MuJoCo Asyn-MTS (b) MuJoCo Syn-MTS (c) USHCN Asyn-MTS (d) NASDAQ Asyn-MTS

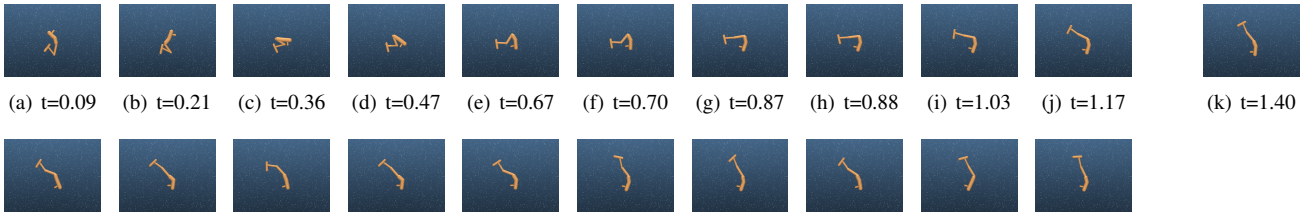
Fig. 8. The observed quantile level v.s. predicted quantile level of three real-world datasets. The dash lines are the baseline models, and the corresponding solid lines are the RFN counterparts. The theoretical best result is the solid black line, and the closer the model is to it, the better the performance is.

the baseline models.

E. Dataset of Climate Records (USHCN)

Variables in natural phenomena, such as climate data, also display strong serial and cross-correlations [38]. For instance, temperature values from one season provide valuable information about temperature patterns in the following season. Additionally, precipitation, such as rain and snow, can cause temperature drops and affect humidity levels. Missing observations are also common in climate data due to inclement weather or equipment malfunctions. In this experiment, we evaluate the performance of the RFN specification on the USHCN dataset.

The United States Historical Climatology Network (USHCN) dataset [27] consists of daily measurements from 1,218 centers across the country. It includes 5 variables: precipitation, snowfall, snow depth, maximum temperature, and minimum temperature. Following the preprocessing approach of [16], we select the training data from the first quarter of the last four years (1996-2000). This yields 4,494 instances, with each instance's time period normalized from 0 to 12.5 at intervals of 0.1. The observations for each variable



(a) t=0.09 (b) t=0.21 (c) t=0.36 (d) t=0.47 (e) t=0.67 (f) t=0.70 (g) t=0.87 (h) t=0.88 (i) t=1.03 (j) t=1.17 (k) t=1.40

Fig. 9. (a)-(j) represent data observed at historical times with unevenly spaced intervals. The objective is to estimate the data distribution at t=1.40. Fig. (k) displays the observed ground truth at t=1.40. We train the Syn-MTS and Asyn-MTS models to forecast the desired distribution at t=1.40. The first and last five figures in the second row display five samples generated from the trained Syn-MTS model (Asyn-MTS model) at t=1.40.

have uneven spacing, making the dataset asynchronous. Consequently, we exclusively employ the Asyn-MTS model to train this dataset.

The results presented in Table III demonstrate the superior performance of the RFN specification over its vanilla counterpart for all baselines when applied to the climate dataset. These findings reinforce that climate data does not conform to a multivariate Gaussian distribution and highlight the importance of capturing the dependence structure among variables for accurate weather forecasting.

F. Dataset of Stock Transactions (NASDAQ)

The capability to forecast stock prices holds importance for investors, offering a strategic advantage in the financial markets [39]. A transaction in the stock market is a match between a buy order and a sell order. Since traders send their buy and sell orders to exchange at random times, the time intervals between transactions are inherently random.

In this section, we utilize the minute-by-minute transaction records of eight biotech stocks from the NASDAQ exchange from July 26, 2016, to April 28, 2017, across 191 trading days [28]. Their ticker symbols are 'BIIB', 'BMRN', 'CELG', 'REGN', 'VRTX', 'GILD', 'INCY', and 'MYL'. As they are companies from the same sector, their stock prices tend to move together, and these movements are strongly correlated.

We partition the 191 trading days into 993 instances, each of which is a 75-minute-long multivariate instance. The dataset contains instances of missing values, making it a multivariate asynchronous time series. Utilizing the Asyn-MTS model, we simultaneously forecast the joint distribution of the stock prices of these eight stocks. Table IV shows that the RFN specification consistently outperforms its non-RFN counterpart in every one of the models.

G. Sensitivity Analysis

In this section, we present additional experiments to analyze our model's performance under various scenarios. In the main paper, we randomly sampled 50% of the values to generate the Syn-MTS and Asyn-MTS datasets. To further test the robustness of our model, we vary the missing rate of the MuJoCo dataset from 25% to 75% to create new Syn-MTS and Asyn-MTS datasets.

The CRPS performance is illustrated in Fig. 10(a) and 10(b), where dashed lines represent the baseline and solid lines represent the RFN specifications. The results indicate that RFN outperforms all baseline models across all missing rates. In addition, we observe that, for the Syn-MTS dataset, performance decreases slightly as the missing rate increases from 25% to 50%, but declines significantly when the missing rate rises from 50% to 75%. This suggests substantial information loss when more than half of the time points are unobserved. In contrast, the Asyn-MTS dataset shows a steady decrease in performance as the missing rate increases, possibly because missing information in one variable can be compensated by observations of other variables.

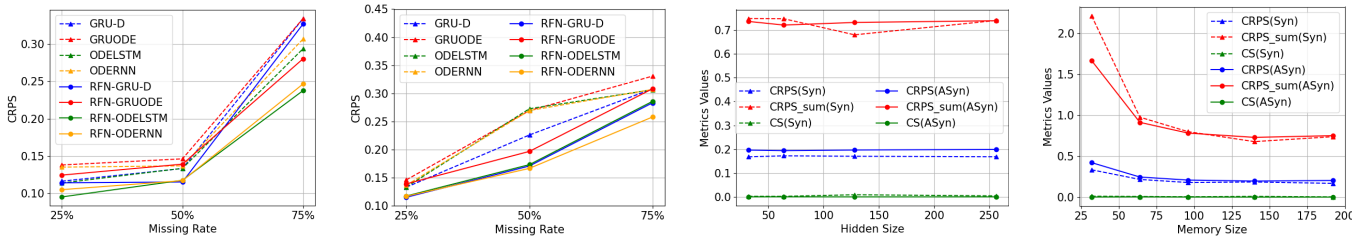
Furthermore, we explore the impact of varying the hidden size of the conditional CNF from 32 to 256, with performance results shown in Fig. 10(c). The performance remains relatively stable across different hidden sizes, indicating that the hidden size of the flow model has minimal influence on the overall performance, demonstrating the robustness of the proposed model. Similarly, we analyze the effect of various memory sizes of the hidden state in the sequential models, as shown in Fig. 10(d). We observe that increasing the hidden state size results in performance improvements due to the model's enhanced ability to capture long-term dependencies. However, as the hidden size increases further, the rate of improvement diminishes.

VI. Discussions & Conclusion

In this paper, we propose an end-to-end learning framework, termed RFN, to address the challenges posed by multivariate irregular time series. The RFN is structured into two core components: a marginal learning layer and a joint learning layer. The marginal learning layer processes the multivariate time series—whether synchronous or asynchronous—by leveraging state-of-the-art sequential models to capture the temporal dynamics at each observed time point. The joint learning layer then models the joint distribution of the variables at each time step using the proposed conditional CNF model.

One of the key innovations of our framework is its ability to overcome the restrictive Gaussian assumption commonly made in time series modeling. By incorporating the dynamic conditional CNF model, our approach facilitates non-parametric learning of the joint distribution at arbitrary continuous-time points, thereby accommodating both temporal and cross-sectional dependencies in complex, irregular time series data.

We extensively evaluate the performance of the RFN framework across several real-world datasets from diverse domains. Our experimental results demonstrate that RFNs consistently



(a) Missing Rate (Syn-MTS) (b) Missing Rate (Asyn-MTS) (c) Hidden Size of Conditional CNF (d) Memory Size of Sequential Model

Fig. 10. The sensitivity analysis across the missing rate, hidden size of conditional CNF (RFN-GRUODE), the memory size of the sequential model (RFN-GRUODE), on Physical Activities (MuJoCo) dataset.

outperform state-of-the-art models that assume Gaussian distributions for modeling dependencies in irregularly sampled time series. These results underscore the robustness and effectiveness of the RFN framework, making it a powerful tool for handling the challenges of irregular time series data in various application domains.

However, we acknowledge several challenges in applying RFN to certain types of datasets. Fisrt, if cross-sectional dependencies are weak, the model's advantage in learning inter-variable relationships diminishes. Second. when applied to datasets with a large proportion of missing values (e.g., greater than 85%), the model's ability to learn reliable dependencies is significantly hindered since extremely sparse data may disrupt temporal and cross-sectional relationships.

Our RFN framework also has some limitations. First, the computational cost of training a CNF-based model is higher than that of simpler Gaussian-based alternatives due to the need for solving continuous-time flow dynamics, see Table VIII. Second, the increased flexibility of normalizing flows introduces additional complexity in optimizing the transformation dynamics, which can lead to unstable gradients and longer training times. To mitigate this issue, we incorporate regularization techniques, such as those proposed in [40], to accelerate convergence and improve stability.

ACKNOWLEDGMENT

Qi WU acknowledges the support from The CityU-JD Digits Joint Laboratory in Financial Technology and Engineering, The Hong Kong Research Grants Council [General Research Fund 11219420/9043008], and The CityU APRC Grant 9610643. The work described in this paper was partially supported by the InnoHK initiative, the Government of the HKSAR, and the Laboratory for AI-Powered Financial Technologies.

REFERENCES

[1] G.-Y. Ban, N. El Karoui, and A. E. Lim, "Machine learning and portfolio optimization," *Management Science*, vol. 64, no. 3, pp. 1136–1154, 2018.

[2] Y. Zhang, Q. Wu, N. Peng, M. Dai, J. Zhang, and H. Wang, "Memory-gated recurrent networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 10 956–10 963.

[3] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella *et al.*, "Deep learning for time series forecasting: Tutorial and literature survey," *ACM Computing Surveys (CSUR)*, 2018.

[4] H. Kremer, S. Gunnemann, and T. Seidl, "Detecting climate change in multivariate time series data by novel clustering and cluster tracing techniques," in *2010 IEEE International Conference on Data Mining Workshops*. IEEE, 2010, pp. 96–97.

[5] P. Gao, X. Yang, R. Zhang, K. Huang, and J. Y. Goulermas, "Explainable tensorized neural ordinary differential equations for arbitrary-step time series prediction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5837–5850, 2022.

[6] S. N. Shukla and B. M. Marlin, "A survey on principles, models and methods for learning from irregularly sampled time series," *arXiv preprint arXiv:2012.00168*, 2020.

[7] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[8] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[9] Y. Luo, X. Cai, Y. Zhang, J. Xu *et al.*, "Multivariate time series imputation with generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[10] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, no. 1, pp. 1–12, 2018.

[11] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating recurrent network training for long or event-based sequences," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[12] M. C. Mozer, D. Kazakov, and R. V. Lindsey, "Discrete event, continuous time rnns," *arXiv preprint arXiv:1710.04110*, 2017.

[13] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[14] Y. Rubanova, R. T. Chen, and D. K. Duvenaud, "Latent ordinary differential equations for irregularly-sampled

time series," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[16] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, "GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] M. Lechner and R. Hasani, "Learning long-term dependencies in irregularly-sampled time series," *arXiv preprint arXiv:2006.04418*, 2020.

[18] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[19] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[20] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank gaussian copula processes," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv preprint arXiv:1711.11053*, 2017.

[22] X. Yan, Q. Wu, and W. Zhang, "Cross-sectional learning of extremal dependence among financial assets," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[23] F. Laio and S. Tamea, "Verification tools for probabilistic forecasts of continuous hydrological variables," *Hydrology and Earth System Sciences*, vol. 11, no. 4, pp. 1267–1277, 2007.

[24] K. Rasul, A.-S. Sheikh, I. Schuster, U. M. Bergmann, and R. Vollgraf, "Multivariate probabilistic time series forecasting via conditioned normalizing flows," in *International Conference on Learning Representations*, 2021.

[25] S. Feng, C. Miao, K. Xu, J. Wu, P. Wu, Y. Zhang, and P. Zhao, "Multi-scale attention flow for probabilistic time series forecasting," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[26] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8857–8868.

[27] M. Menne, C. Williams Jr, and R. Vose, "Long-term daily and monthly climate records from stations across the contiguous united states," *Website http://cdiac. ornl. gov/epubs/ndp/ushcn/access. html [accessed 23 September 2010]*, 2010.

[28] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 2627–2633.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[30] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[31] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1530–1538.

[32] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[33] J. E. Matheson and R. L. Winkler, "Scoring rules for continuous probability distributions," *Management Science*, vol. 22, no. 10, pp. 1087–1096, 1976.

[34] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2796–2804.

[35] P. Glasserman, *Monte Carlo methods in financial engineering*. Springer, 2004, vol. 53.

[36] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973.

[37] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[38] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Deep air quality forecasting using hybrid deep learning framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 6, pp. 2412–2424, 2019.

[39] Y. Zhao, H. Du, Y. Liu, S. Wei, X. Chen, F. Zhuang, Q. Li, and G. Kou, "Stock movement prediction based on bi-typed hybrid-relational market knowledge graph via dual attention networks," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[40] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. Oberman, "How to train your neural ode: the world of jacobian and kinetic regularization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3154–3164.

[41] S. N. Shukla and B. M. Marlin, "Interpolation-prediction networks for irregularly sampled time series," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=r1efr3C9Ym

[42] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, "Patient subtyping via time-aware lstm networks," in *Proceedings of the 23rd ACM SIGKDD International Vonference on Knowledge Discovery and*

Data Mining, 2017, pp. 65–74.

[43] T. Gneiting, F. Balabdaoui, and A. E. Raftery, "Probabilistic forecasts, calibration and sharpness," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 69, no. 2, pp. 243–268, 2007.

Yijun Li received the Ph.D. degree in data science from the City University of Hong Kong, Hong Kong, in 2024. He is currently a postdoctoral researcher at the Laboratory for AI-Powered Financial Technologies Limited, Hong Kong. His research interests include time series forecasting, causal inference, and financial technology.

Cheuk Hang Leung received the BSc. degree in mathematics from the University of Hong Kong, in 2009, the MSc degree from the Hong Kong University of Science and Technology, in 2012, and the PhD degree from the Chinese University of Hong Kong, in 2018. He is now working at the City University of Hong Kong JD Digits Joint Laboratory in Financial Technology and Engineering.

Qi Wu received the Ph.D. degree in applied mathematics from Columbia University, New York, NY, USA, in 2013. He is currently an Associate Professor with the School of Data Science, City University of Hong Kong (CityU), Hong Kong. He is also the Program Leader at Laboratory for AI-Powered Financial Technologies Ltd., Hong Kong. Prior to CityU, he was an Assistant Professor at The Chinese University of Hong Kong and a Finance Practitioner at Lehman Brothers, London, U.K.; UBS, Stamford, CT, USA; and DTCC, New York.