PROFITI: PROBABILISTIC FORECASTING OF IRREGULAR TIME SERIES VIA CONDITIONAL FLOWS

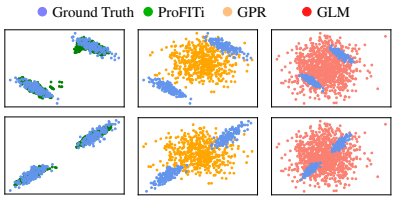
Anonymous authors
Paper under double-blind review

Probabilistic forecasting of irregularly sampled multivariate time series with missing values is an important problem in many fields, including astronomy, finance, and healthcare. Traditional methods for this task often rely on differential equations based models and make an assumption on the target distribution. In recent years, normalizing flow models have emerged as a promising approach for density estimation and uncertainty quantification, offering a flexible framework that can capture complex dependencies. In this work, we propose a novel model ProFITi for probabilistic forecasting of irregular time series with missing values using conditional normalizing flows. In this approach, the model learns a joint probability distribution over the future values of the time series conditioned on the past observations and query (future) time-channel information. As components of our model, we introduce a novel invertible triangular attention layer, and an invertible non-linear activation function on and onto the whole real line. We conduct extensive experiments on 3 datasets, and demonstrate that the proposed model, ProFITi, provides significantly better forecasting likelihoods compared to the existing baseline models. Specifically, on an average ProFITi provides 4 times better likelihood over the previously best model.

1 INTRODUCTION

Irregularly sampled multivariate time series with missing values (IMTS) is common in various real-world scenarios such as astronomy, health, and finance. While accurate forecasting of these IMTS is important for informed decision-making, estimating the uncertainty associated with these forecasts becomes crucial to avoid overconfidence. Ordinary Differential Equations (ODE) based models are widely used for the task. Besides the low computational efficiency, the current ODE-based models (Schirmer et al., 2022; De Brouwer et al., 2019; Biloš et al., 2021) offer only marginal likelihood estimates. In practical applications, joint distribution is desired to capture dependencies and study forecasting scenarios or trajectories.

In this study, we propose two novel components that can be used in flow models: a sorted invertible triangular attention layer (`SITA`) parametrized by conditioning input, and an invertible activation function, `Shiesh`, that is on and onto $\mathbb{R}$. We further propose a novel conditional flow model called `ProFITi`, for **Pro**babilistic **F**orecasting of **I**rregularly sampled Multivariate **Ti**me series ProFITi is a permutation invariant model and designed to learn conditional permutation invariant structured distributions. It consists of several invertible blocks build using SITA and Shiesh functions. Being a Flow-based model, ProFITi can learn any random conditional joint distribution, we demonstrate this with mixture of Gaussian in Figure 1.



(a) ProFITi (b) GPR (c) GLM

Figure 1: (a) Distribution generated by ProFITi, (b) Gaussian Process Regression and (c) Generalized Linear Model. ProFITi provides a distribution close to the ground truth. See Section F.

Through extensive experiments, we demonstrate that our ProFITi achieves state-of-the-art probabilistic forecasting results for IMTS. Our contributions are as follows:

1. To the best of our knowledge, we are the first to investigate **normalizing flow based models for conditional permutation invariant structured distributions**. This makes them usable for probabilistic IMTS forecasting tasks.
2. We provide a novel invertible equivariant transformation, making the self attention mechanism invertible (in the last column), **sorted invertible triangular self attention**.

Table 1: Summary of Important models that 1. can be applied to Time Series with irregular sampling (Irreg. Samp.), or missing values (Miss. Vals.), 2. can predict marginal distributions (Marg. Dist.) or joint distributions (Joint Dist.), 3. can learn on conditional densities (Condition), 4. density of sequences with variable lengths (Dynamic) or 5. Permutation Invariant (Perm. Inv.).

Model	Irreg. Samp.	Miss. Vals.	Marg. Dist.	Joint Dist.	Condition	Dynamic	Perm. Inv.
GRU-ODE (De Brouwer et al., 2019)	✓	✓	(Param)	×	✓	✓	✓
Neural Flows (Biloš et al., 2021)	✓	✓	(Param)	×	✓	✓	✓
CRU (Schirmer et al., 2022)	✓	✓	(Param)	×	✓	✓	✓
GPR (Dürichen et al., 2015)	✓	✓	(Param)	(Param)	✓	✓	✓
GraFITi (Yalavarthi et al., 2023)	✓	✓	×	×	✓	✓	✓
RealNVP (Dinh et al., 2017)	×	×	×	✓	×	×	×
Inv. Autoreg (Kingma et al., 2016)	×	×	×	✓	×	×	×
Selv. Flow (van den Berg et al., 2018)	×	×	×	✓	×	×	×
Residual Flow (Behrmann et al., 2019)	×	×	×	✓	×	×	×
Graphical (Wehenkel & Louppe, 2021)	×	×	×	✓	×	×	×
Cond. NF Winkler et al. (2019)	×	×	×	✓	✓	×	×
Attn. Flow Sukthanker et al. (2022)	×	×	×	✓	✓	×	×
Inv. Dot. Attn Zha et al. (2021)	×	×	×	✓	✓	×	×
E(N) (Satorras et al., 2021)	×	×	×	✓	×	✓	✓
GNF (Liu et al., 2019)	×	×	×	✓	×	✓	✓
SNF (Biloš & Günnemann, 2021)	×	×	×	✓	×	✓	✓
MAF (Rasul et al., 2021)	×	×	✓	×	✓	✓	✓
CTFP (Deng et al., 2020)	✓	×	✓	×	✓	✓	✓
NKF (de Bézenac et al., 2020)	×	×	×	✓	✓	✓	✓
QFR (Si et al., 2022)	×	×	×	✓	✓	✓	✓
ProFITi (ours)	✓	✓	✓	✓	✓	✓	✓

3. We provide a novel non-linear, invertible, differentiable activation function on and onto the whole real line, **Shiesh**. This activation function can be used in normalizing flows.
4. We provide a normalizing flow based model **ProFITi** for probabilistic forecasting of IMTS build from invertible self attention layers and transformation layers using Shiesh activation.
5. We conduct extensive experiments on 3 real-world IMTS datasets, and show that our proposed model, ProFITi, significantly outperforms baselines in terms of normalized joint negative log-likelihood.

Implementation code: `https://anonymous.4open.science/r/ProFITi-8707/`.

2 Literature Review

Probabilistic Forecasting Models for IMTS Probabilistic IMTS forecasting often relies on variational inference or predicting distribution parameters. Neural ODE models (Chen et al., 2018) are popular, with VAE-based variants combining probabilistic latent states with deterministic networks. Other approaches like latent-ODE (Rubanova et al., 2019), GRU-ODE-Bayes (De Brouwer et al., 2019), Neural-Flows Biloš et al. (2021), and Continuous Recurrent Units (CRUs) (Schirmer et al., 2022) have shown accurate results but provide only marginal distributions, lacking joint distributions. In contrast, Gaussian Process Regression (GPR) models (Dürichen et al., 2015; Li & Marlin, 2015; 2016; Bonilla et al., 2007) offer full joint posterior distributions for forecast outputs. However, multitask GPR can struggle due to positive definite covariance matrix constraints and computational demands on long time series data due to the matrix inverting operations.

Normalizing Flows for variable input size We deal with predicting distribution for variable length targets. This utilizes equivariant models for a dataset comprised of sets, as shown in (Biloš & Günnemann, 2021; Satorras et al., 2021; Liu et al., 2019). All the models apply continuous normalizing flows which require solving an ODE driven by a neural network. They tend to be slow due to the numerical integration process. Additionally, they cannot incorporate conditioning inputs.

Conditioning Normalizing Flows The modeling of conditional data densities has been extensively explored within the realm of computer vision literature (Khorashadizadeh et al., 2023; Winkler et al., 2019; Anantha Padmanabha & Zabaras, 2021). They involve applying basic normal-

izing flow blocks such as affine transformations (Dinh et al., 2017), autoregressive transformations (Kingma et al., 2016) or Sylvester flow blocks (van den Berg et al., 2018). Often the conditioning values are appended to the target while passing through the flow layers as demonstrated in (Winkler et al., 2019). For continuous data representations a few works are proposed (Kumar et al., 2020; de Bézenac et al., 2020; Rasul et al., 2021; Si et al., 2022). However, all the methods deal with regular multivariate time series and cannot handle IMTS.

Flows with Invertible Attention To the best of our knowledge, there has been only two works that apply invertible attention in Normalizing Flows. Sukthanker et al. (2022) proposed invertible attention mechanism by adding the identity matrix to softmax attention. However, the major disadvantage of this mechanism is that softmax yields only positive values in the attention matrix and does not allow learning negative covariances. Zha et al. (2021) introduced residual attention similar to invertible residual flow (Chen et al., 2019; Behrmann et al., 2019). It has similar problems related to residual flows (Behrmann et al., 2019) such as lack of analytical inverse making it extremely slow to infer due to requirement of numerical methods. Additionally, these attention mechanisms often have problems with computing the determinant of the attention matrix as it has a complexity of $\mathcal{O}(K^3)$ with K being the sequence length. We provide the summary of the related work in Table 1.

3 Problem Setting & Analysis

The IMTS Forecasting Problem. An **irregularly sampled multivariate times series with missing values** (called briefly just **IMTS** in the following), is a finite sequence $x^{\text{obs}} = \left((t_i^{\text{obs}}, c_i^{\text{obs}}, o_i^{\text{obs}})\right)_{i=1:I}$ of unique triples, where $t_i^{\text{obs}} \in \mathbb{R}$ denotes the time, $c_i^{\text{obs}} \in \{1, ..., C\}$ the channel and $o_i^{\text{obs}} \in \mathbb{R}$ the value of an observation, $I \in \mathbb{N}$ the total number of observations across all channels and $C \in \mathbb{N}$ the number of channels. Let $\text{TS}(C) := (\mathbb{R} \times \{1, \dots, C\} \times \mathbb{R})^*$ denote the space of all IMTS with C channels.[1]

An **IMTS query** is a finite sequence $x^{\text{qu}} = \left((t_k^{\text{qu}}, c_k^{\text{qu}})\right)_{k=1:K} \in \text{Q}(C) := (\mathbb{R} \times \{1, \dots, C\})^*$ of just time points and channels (also unique), a sequence $y \in \mathbb{R}^K$ we call an answer and denote by $\text{QA}(C) := \{(x^{\text{qu}}, y) \in \text{Q}(C) \times \mathbb{R}^* \mid |x^{\text{qu}}| = |y|\}$ the space of all queries and possible answers.

The **IMTS probabilistic forecasting problem** then is, given a dataset $\mathcal{D}^{\text{train}} := \left((x^{\text{obs}}, x^{\text{qu}}, y)\right)^* \in \text{TS}(C) \times \text{QA}(C)$ of triples of time series, queries and answers from an unknown distribution p (with $\min_k t_{n,k}^{\text{qu}} > \max_i t_{n,i}^{\text{obs}}$), to find a model $\hat{p}$ that maps each observation/query pair $(x^{\text{obs}}, x^{\text{qu}})$ to a joint density over answers, $\hat{p}(y_1, \dots, y_K \mid x^{\text{obs}}, x^{\text{qu}})$, such that the expected negative log likelihood is minimal:

$$\ell^{\text{NLL}}(\hat{p}; p) := -\mathbb{E}_{(x^{\text{obs}}, x^{\text{qu}}, y) \sim p} \log \hat{p}(y \mid x^{\text{obs}}, x^{\text{qu}})$$

Please note, that the number C of channels is fixed, but the number I of past observations and the number K of future observations queried may vary over instances $(x^{\text{obs}}, x^{\text{qu}}, y)$. If query sizes K vary, instead of (joint) negative log likelihood one also can normalize by query size to make numbers comparable over different series and limit the influence of large queries, the **normalized joint negative log likelihood** NJNL:

$$\ell^{\text{NJNL}}(\hat{p}; p) := - \mathop{\mathbb{E}}_{(x^{\text{obs}}, x^{\text{qu}}, y) \sim p} \frac{1}{|y|} \log \hat{p}(y \mid x^{\text{obs}}, x^{\text{qu}}) \tag{1}$$

Problem Analysis and Characteristics. As the problem is not just an (unconditioned) density estimation problem, but the distribution of the outputs depends on both, the past observations and the queries, a **conditional density model** is required (**requirement 1**).

A crucial difference from many settings addressed in the related work is that we look for probabilistic models of the **joint distribution** of all queried observation values $y_1, \dots, y_K$, not just at the **single variable marginal distributions** $p(y_k \mid x^{\text{obs}}, x_k^{\text{qu}})$ (for $k = 1{:}K$). The problem of marginal distributions is a special case of our formulation where all queries happen to have just one element (always $K = 1$).

[1] we use * to indicate a finite arbitrary natural number

So for joint probabilistic forecasting of IMTS, models need to output densities on a **variable number of variables** (**requirement 2**).

Furthermore, whenever two query elements get swapped, a generative model should swap its output accordingly, a density model should yield the same density values for outputs swapped the same way, i.e., the model should be **permutation invariant** (**requirement 3**): for every permutation π:

$$\hat{p}(y_1, \ldots, y_K \mid x^{\text{obs}}, x_1^{\text{qu}}, \ldots, x_K^{\text{qu}}) = \hat{p}(y_{\pi(1)}, \ldots, y_{\pi(K)} \mid x^{\text{obs}}, x_{\pi(1)}^{\text{qu}}, \ldots, x_{\pi(K)}^{\text{qu}}) \tag{2}$$

4 Invariant Conditional Normalizing Flow Models

Normalizing flows. While parametrizing a specific distribution such as the Normal, is a simple and robust approach to probabilistic forecasting that can be added on top of any point forecasting model (for marginal distributions or fixed-size queries at least), such models are less suited for targets having a more complex distribution. Then typically normalizing flows are used (Rippel & Adams, 2013; Papamakarios et al., 2021). A normalizing flow is an (unconditional) density model for variables $y \in \mathbb{R}^K$ consisting of a simple base distribution, typically a standard normal $p_Z(z) := \mathcal{N}(z; 0_K, \mathbb{I}_{K\times K})$, and an invertible, differentiable, parametrized map $f(z;\theta)$; then

$$\hat{p}(y;\theta) := p_Z(f^{-1}(y;\theta)) \left| \det \left(\frac{\partial f^{-1}(y;\theta)}{\partial y} \right) \right| \tag{3}$$

is a proper density, i.e., integrates to 1, and can be fitted to data minimizing negative log likelihood via gradient descent algorithms. A normalizing flow can be conditioned on predictor variables $x \in \mathbb{R}^M$ by simply making f dependent on predictors x, too: $f(z; x, \theta)$. f then has to be invertible w.r.t. z for any x and θ (Trippe & Turner, 2018).

Invariant conditional normalizing flows. A conditional normalizing flow represents an invariant conditional distribution in the sense of eq. 2, if i) its predictors x also can be grouped into K elements $x_1, \ldots, x_K$ and possibly common elements x^{com}: $x = (x_1, \ldots, x_K, x^{\text{com}})$, and ii) its transformation f is equivariant in stacked $x_{1:K}$ and z:

$$f(z^\pi; x_{1:K}^\pi, x^{\text{com}}, \theta)^{\pi^{-1}} = f(z; x_{1:K}, x^{\text{com}}, \theta) \quad \forall \text{permutations } \pi$$

where $z^\pi := (z_{\pi(1)}, \ldots, z_{\pi(K)})$ denotes a permuted vector. We call this an **invariant conditional normalizing flow model**. If K is fixed, we call it **fixed size**, otherwise **dynamic size**.

Invariant conditional normalizing flows via continuous flows. Invariant conditional normalizing flow models have been developed in the literature based on the **continuous flow** approach (Chen et al., 2018; Grathwohl et al., 2019), where the transformation f is specified implicitly by an ordinary differential equation with time-dependent vector field $g : \mathbb{R} \times \mathbb{R}^M \to \mathbb{R}^M$:

$$f^{-1}(z) := v(1) \quad \text{with } v : [0,1] \to \mathbb{R}^K \text{ being the solution of } \frac{\partial v}{\partial \tau} = g(\tau, v(\tau)), \quad v(0) := z \tag{4}$$

τ often is called virtual time to clearly distinguish it from time as an input variable. The vector field g is represented by a parametrized function $g(\tau, v; \theta)$ and then can be learnt. Continuous flow models can be made conditional by simply adding the predictors x to the inputs of the vector field, too: $g(\tau, v; x, \theta)$. Unconditional structured continuous flow models can be made permutation invariant by simply making the vector field permutation equivariant (Köhler et al., 2020; Li et al., 2020; Biloš & Günnemann, 2021): $g(\tau, v^\pi; \theta)^{\pi^{-1}} = g(\tau, v; \theta)$. To make *conditional* structured continuous flow models permutation invariant, the vector field has to be **jointy permutation equivariant** in outputs v and predictors x:

$$g(\tau, v^\pi; x^\pi, \theta)^{\pi^{-1}} = g(\tau, v; x, \theta)$$

The primary choice for a dynamic size, equivariant, parametrized function is self attention (SA):

$$A(X) := XW_{\text{Q}}(XW_{\text{K}})^T, \quad A^{\text{softmax}}(X) := \text{softmax}(A(X))$$
$$\text{SA}(X) := A^{\text{softmax}}(X) \cdot XW_{\text{V}}$$

where X is a matrix containing the elements $x_{1:K}$ as rows, $W_\mathrm{Q}, W_\mathrm{K}, W_\mathrm{V}$ are parameter matrices (not depending on the number of rows of X) and the softmax is taken rowwise.

Self attention has been used in the literature as is for unconditional vector fields (Köhler et al., 2020; Li et al., 2020; Biloš & Günnemann, 2021). To be used in a conditional vector field, X will have to contain both, the condition elements $x_{1:K}$ and the base samples $z_{1:K}$ stacked:

$$X := \begin{bmatrix} x_1^T & z_1 \\ \vdots & \vdots \\ x_K^T & z_K \end{bmatrix}$$

Invariant conditional normalizing flows via invertible self attention. When using self attention as vector field inside a continuous flow as in the previous section, then the continuous flow will provide invertibility. While an elegant and generic approach, continuous flows require ODE solvers and have been reported to be brittle and not straight-forward to train. We develop an alternative idea: to make self attention itself invertible (in the last column of X, which contains z). Then it can be used directly, without any need for an ODE wrapper. To get **invertible self attention (ISA)** (in the last column), we i) fix the last row of attention query and key matrices W_Q and W_K to zero, in effect *computing the attention matrix A on the conditioners $x_{1:K}$ alone*, ii) fix all but the last rows of W_V to zero and its last row to all ones, in effect *using the base sample $z_{1:K}$ alone as attention value*, and iii) regularize the attention matrix A sufficiently to become invertible:

$$A^\mathrm{reg}(X) := \frac{1}{\|A(X)\|_2 + \epsilon} A(X) + \mathbb{I} \tag{5}$$

$$\mathrm{ISA}(X) := A^\mathrm{reg}(X_{:,1:|X|-1}) X_{:,|X|} \tag{6}$$

$\epsilon > 0$ a small constant. We note that different from a simple linear flow, the slope matrix A^reg is not a parameter of the model, but computed from the conditioners $x_{1:K}$. Our approach is different from iTrans attention (Sukthanker et al., 2022, fig. 17) that makes attention invertible more easily via $A^\mathrm{iTrans} := A^\mathrm{softmax}(X) + \mathbb{I}$ using the fact that spectral radius $\sigma(A^\mathrm{softmax}(X)) \leq 1$, but therefore is restricted to non-negative interaction weights.

The attention matrix A^reg will be dense in general and thus slow to invert, taking $\mathcal{O}(K^3)$ operations. Following ideas for autoregressive flows and coupling layers, a triangular slope matrix would allow a much more efficient inverse pass, as its determinant can be computed in $\mathcal{O}(K)$ and linear systems can be solved in $\mathcal{O}(K^2)$. This does not restrict the expressivity of the model, as due to the Knothe–Rosenblatt rearrangement (Villani, 2009) from optimal transport theory, any two probability distributions on $\mathbb{R}^K$ can be transformed into each other by flows with a locally triangular Jacobian (where the local Jacobians in ODE models of optimal transport correspond to layers in the transformation of a normalized flow). Unfortunately, just masking the upper triangular part of the matrix will destroy the equivariance of the model. We resort to the simplest way to make a function equivariant: we sort the inputs before passing them into the layer and revert the outputs to the original ordering. We call this approach **sorted invertible triangular self attention (SITA)**:

$$\begin{aligned} \pi &:= \mathrm{argsort}(x_1 S, \ldots, x_K S) \\ A^\mathrm{tri}(X) &:= \text{softplus-diag}(\text{lower-triangular}(A(X))) + \epsilon\mathbb{I} \\ \mathrm{SITA}(X) &:= (A^\mathrm{tri}(X^\pi_{:,1:|X|-1}) X^\pi_{:,|X|})^{\pi^{-1}} \end{aligned}$$

where π operates on the rows of X. Sorting is a simple lexicographic sort along the dimensions of the $x_k S$. The matrix S allows to specify a sorting criterion, e.g., a permutation matrix. For example, we will sort time series queries first by time stamp, so that the triangular matrix corresponds to a causal attention, second by channel, fixing some order of the channels. method.

5 A New Activation Function for Normalizing Flows

The transformation function f of a normalizing flow usually is realized as a stack of several simple functions. As in any other neural network elementwise applications of a function, called activation functions, is one of those layers that allows for non-linear transformations. However, most of the common activation functions used in deep learning such as ReLU are not applicable for normalizing

flows, because they are not invertible (E1). Some like ELU cannot be used throughout the layer stack, because their output domain $\mathbb{R}^+$ does not cover all real numbers (E2). Some like Tanh-shrink $(\text{tanhshrink}(u) := u - \tanh(u))$ are invertible and cover the whole real line, but they have a zero gradient somewhere (for Tanh-shrink at 0) that will make computing the inverse of the normalizing factor $\left|\det\left(\frac{\partial f(u)}{\partial u}\right)\right|$ for the normalizing flow impossible (E3).

To be used as a standalone layer in a normalizing flow, an activation function must fulfill these three requirements: **E1**. be invertible, **E2**. cover the whole real line and **E3**. have no zero gradients. Out of all activation functions in the pytorch library only Leaky-ReLU and P-ReLU meet all three requirements (see table 2). Both, Leaky-ReLU and P-ReLU usually are used with a slope on their negative branch being well less than 1, so that stacking many of them might lead to small gradients also causing problems for the normalizing constant of a normalizing flow.

Table 2: Properties of existing activation functions.

Activation	E1	E2	E3
ReLU	×	×	×
Leaky-ReLU	✓	✓	✓
P-ReLU	✓	✓	✓
ELU	✓	×	✓
SELU	✓	×	✓
Swish	×	×	×
GELU	×	×	✓
Tanh	✓	×	✓
Sigmoid	✓	×	✓
Tanh-shrink	✓	✓	×
Shiesh	✓	✓	✓

Unconstrained monotonic neural networks (Wehenkel & Louppe, 2019) have been proposed as versatile, learnable activation functions for normalizing flows, being basically a continuous normalizing flow for each scalar variable u separately and a scalar field g implemented by a neural network:

$$a(u) := v(1) \quad \text{with } v : [0,1] \to \mathbb{R} \text{ being the solution of } \frac{\partial v}{\partial \tau} = g(\tau, v(\tau)), \quad v(0) := u \tag{7}$$

Lemma 2 (Section D) establishes the equivalence of unconstrained monotonic neural networks with continuous normalizing flows.

In consequence, they suffer from the same issues as any continuous normalizing flow: they are slow as they require explicit integration of the underlying ODE. Besides requirements E1–E3, activation functions will profit from further desired properties: **D1**. having an analytic inverse, **D2**. having an analytic Jacobian and **D3**. having a bounded gradient. Unconstrained Monotonic Neural Networks do not have desired properties D1 and D2 and provide no guarantees for property D3.

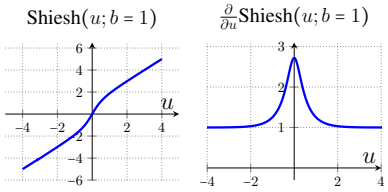


Figure 2: (left) Shiesh function, (right) partial derivative.

Instead of parametrizing the scalar field g and learn it from data, we make an educated guess and choose a specific function with few parameters for which eq. 7 becomes explicitly solvable and requires no numerics at runtime: for the scalar field $g(\tau, a; b) := \tanh(b \cdot a(\tau))$ the resulting ODE

$$\frac{\partial v}{\partial \tau} = \tanh(b \cdot v(\tau)), \quad v(0) := u$$

has an explicit solution (Section E.1)

$$v(\tau; u, b) = \frac{1}{b} \sinh^{-1}\left(e^{b\cdot\tau} \cdot \sinh(b \cdot u)\right)$$

yielding our activation function Shiesh:

$$\text{Shiesh}(u; b) := a(u) := v(1; u, b) = \frac{1}{b} \sinh^{-1}\left(e^{b} \cdot \sinh(b \cdot u)\right) \tag{8}$$

being invertible, covering the whole real line and having no zero gradients (E1–E3) and additionally with analytical inverse and gradient (D1 and D2)

$$\text{Shiesh}^{-1}(u; b) = \frac{1}{b} \sinh^{-1}\left(e^{-b} \cdot \sinh(b \cdot u)\right) \tag{9}$$

$$\frac{\partial}{\partial u}\text{Shiesh}(u; b) = \frac{e^b \cosh(b \cdot u)}{\sqrt{1 + \left(e^b \sinh(b \cdot u)\right)^2}}$$

and bounded gradient (D3) in the range $(1, e^b]$ (Section E.4). Fig. 2 shows a function plot. In our experiments we fixed its parameter $b = 1$.

6 Overall ProFITi Model Architecture

Invertible attention and the Shiesh activation function model, in a systematic way, inter-dependencies between variables and non-linearity respectively, but do not move the zero point. To accomplish the latter, we use a third layer called **element wise linear transformation layer** (EL):

$$\mathrm{EL}(y_k; x_k) := y_k \cdot \mathrm{NN}^{\mathrm{sca}}(x_k) + \mathrm{NN}^{\mathrm{trs}}(x_k) \quad (10)$$

where $\mathrm{NN}^{\mathrm{sca}}$ and $\mathrm{NN}^{\mathrm{trs}}$ are neural networks for scaling and translation. $\mathrm{NN}^{\mathrm{sca}}$ is equipped with a $\exp(\tanh)$ output function to make it positive and bounded, guaranteeing inverse. We combine all three layers from eq. 6, 8, and 10 to a block

$$\text{profiti-block}(y; x) := \mathrm{Shiesh}(\mathrm{EL}(\mathrm{ISA}(z; x); x)) \quad (11)$$

We add a transformation layer with slope fixed to 1 as initial encoding on the y-side of the model. See fig. 3 for an overview of its architecture.

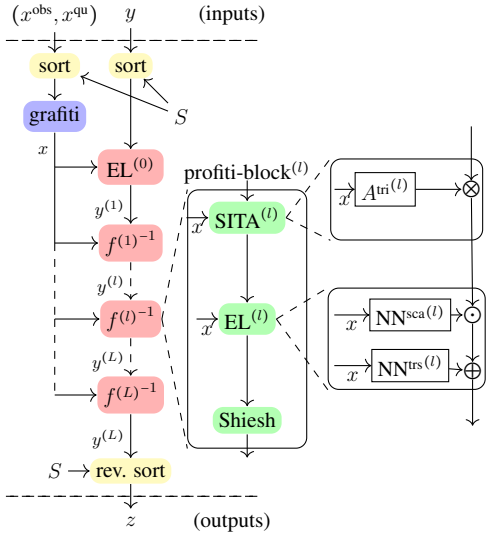


Figure 3: ProFITi architecture. $\otimes$: dot product, $\odot$: Hadamard product, and $\oplus$: addition.

Query embedding. As discussed in Section 4, for probabilistic time series forecasting we have to condition on both, the past observations x^{obs} and the time point/channel pairs x^{qu} of interest that are queried. While in principle any equivariant encoder could be used, an encoder that leverages the relationships between those two pieces of the conditioner is crucial. We use GraFITi (Yalavarthi et al., 2023), a graph based deterministic equivariant forecasting model for IMTS that provide state-of-the-art performance as encoder

$$(x_1, \ldots, x_K) := \mathrm{GraFITi}(x_1^{\mathrm{qu}}, \ldots, x_K^{\mathrm{qu}}, x^{\mathrm{obs}})$$

The encoded conditioners $x_1, \ldots, x_K$ then are fed into ProFITi as in eq. 11. The Grafiti encoder is trained end-to-end within the Profiti model, we did not pretrain it.

Training. We train the ProFITi model $\hat{p}$ for the normalized joint negative log-likelihood loss (NJNL; eq. 1) which is written in terms of the transformation $f^{-1}(\cdot;\cdot;\theta)$ of the normalizing flow and its parameters θ yields:

$$\ell^{\mathrm{NJNL}}(\theta) := \ell^{\mathrm{NJNL}}(\hat{p}; p) \quad (12)$$
$$= - \mathop{\mathbb{E}}_{(x^{\mathrm{obs}}, x^{\mathrm{qu}}, y) \sim p} \frac{1}{|y|} \left(\sum_{k=1}^{|y|} p_Z(f^{-1}(y; x^{\mathrm{obs}}, x^{\mathrm{qu}}; \theta)_k) - \log \left| \det \left(\frac{\partial f^{-1}(y; x^{\mathrm{obs}}, x^{\mathrm{qu}}; \theta)}{\partial y} \right) \right| \right)$$

7 Experiments

7.1 Experiment for Normalized Joint Negative Log-likelihood

Datasets We use 3 publicly available real-world medical IMTS datasets: MIMICIII (Johnson et al., 2016), MIMICIV (Johnson et al., 2021), and Physionet (Silva et al., 2012), for evaluating the ProFITi architecture. Datasets contain ICU patient records collected over 48 hours. The preprocessing procedures outlined in (Yalavarthi et al., 2023; Scholz et al., 2023; Biloš et al., 2021; De Brouwer et al., 2019) were applied to MIMIC-III, and MIMIC-IV datasets. Consequently, observations in MIMIC-III and MIMIC-IV were rounded to intervals of 30 minutes and 1 min, respectively. As for Physionet'12, the dataset was processed according to (Yalavarthi et al., 2023; Che et al., 2018; Cao et al., 2018; Tashiro et al., 2021) and obtain hourly observations.

Baseline Models ProFITi is compared with 3 probabilistic IMTS forecasting models: CRU (Schirmer et al., 2022), Neural-Flows Biloš et al. (2021), GRU-ODE-Bayes De Brouwer et al.

Table 3: Normalized Joint Negative Log-likelihood (NJNL), lower the best. Best is presented in bold. OOM indicates out of memory error. Likelihood ↑ shows percentage gain and actual gain in Normalized Joint Likelihood $e^{-\text{NJNL}}$ (not log-level NJNL) compared to the next best model.

	Physioinet'12	MIMIC-III	MIMIC-IV
GPR	1.367±0.074	3.146±0.359	2.789±0.057
HETAVE	0.561±0.012	0.794±0.032	OOM
GRU-ODE	0.501±0.001	0.837±0.012	0.823±0.318
Neural-Flows	0.496±0.001	0.835±0.014	0.689±0.087
CRU	0.741±0.001	1.090±0.001	OOM
CNF	1.057±0.007	1.123±0.005	1.041±0.010
GraFITi+	0.367±0.021	0.695±0.019	0.287±0.040
ProFITi (ours)	**-0.766±0.038**	**-0.240±0.068**	**-1.856±0.051**
Likelihood ↑	210% (3.1×)	150% (2.5×)	755% (8.5×)

(2019). We also extend the state-of-the-art deterministic forecasting model GraFITi (Yalavarthi et al., 2023) for the probabilistic setup by learning variance of the normal distribution along with the mean and call it as GraFITi+. Because one can often use interpolation models for the forecasting task, we include HETVAE, state-of-the-art probabilistic interpolation model, for the comparison. Further, we apply Multi-task Gaussian Process Regression model (GPR) (Dürichen et al., 2015). Since, we could not find any CNF model that works for dynamic size and conditioning input, we adapt continuous normalizing flow model (CNF) from (Biloš & Günnemann, 2021) to make it conditional as explained in Section 4. We set L2 regularized multivariate attention as g in eq. 4.

Protocol We split the dataset into Train, Validation and Test in 70:10:20 ratio respectively. We select the hyperparameters using validation dataset on 10 random hyperparameter sets. We run for 5 iterations with random seeds on the test dataset with the chosen hyperparameters. Following (Biloš et al., 2021; De Brouwer et al., 2019; Yalavarthi et al., 2023), we set observation range as the first 36 hours and forecast the next 3 time steps. All the models were experimented using the PyTorch library on GeForce RTX-3090 and 1080i GPUs. We compare the models for Normalized Joint Negative Log-likelihood (NJNL) loss (eq. 1). Except for GPR, CNF and ProFITi, we take the average of the marginal negative log-likelihoods of all the observations in a series to compute NJNL for that series.

Results Table 3 demonstrates the Mean Normalized Joint Negative Log-likelihood for all the datasets (lower the best). Best results are presented in bold. ProFITi outperforms all the prior approaches with significant margin on all the three datasets. The next best performing model in Physionet'12 and MIMIC-III dataset is GraFITi+ whereas for MIMIC-IV it is NeuralFlows. We note that although GPR considers joint likelihoods, it performs poorly because of having very few parameters. We cannot run CRU model for MIMIC-IV in our GPU with 48GB memory as it is giving memory errors. Surprisingly, despite being a flow model, CNF did not perform as well in our tasks. *The performance gains shown in the table are not on NJNL but normalized joint likelihoods ($e^{\text{-NJNL}}$), because of* 0 *in the space of NJNL.*

7.2 Auxiliary Experiment for Marginals and Point forecast

Existing models (De Brouwer et al., 2019; Biloš et al., 2021) cannot predict joint distributions hence their evaluation is restricted to Marginal Negative Log-likelihood MNL:

$$\ell^{\text{MNL}}(\hat{p}; \mathcal{D}^{\text{test}}) := -\frac{1}{\sum_{(x^{\text{obs}}, x^{\text{qu}}, y) \in D^{\text{test}}} |y|} \sum_{(x^{\text{obs}}, x^{\text{qu}}, y) \in D^{\text{test}}} \sum_{k=1}^{|y|} \log\left(\hat{p}\left(y_k \mid x^{\text{obs}}, x^{\text{qu}}\right)\right) \tag{13}$$

For additional comparison with published results of the baselines, we evaluate ProFITi for MNL and MSE as well. Table 4 compares ProFITi with GRU-ODE, Neural-Flows, and GraFITi+. Results also include NeuralODE-VAE (Chen et al., 2018), Sequential-VAE (Krishnan et al., 2015; 2017) and GRU-D (Che et al., 2018) from the published sources. To compute MNL, we modify A^{tri} of ProFITi by zeroing off-diagonal elements after training. For point estimates for MSE, we sample 100 values from profiti together with their density values and output the one with the highest density value.

Table 4: Results for auxiliary experiments, Marginal Negative Log-likelihood (MNL), Mean Squared Error (MSE), lower the best. Comparing ProFITi with published results (in brackets): † from De Brouwer et al. (2019), ‡ from Biloš et al. (2021), and ♯ from Yalavarthi et al. (2023).

	MIMIC-III		MIMIC-IV	
	MNL	MSE	MNL	MSE
NeuralODE-VAE	(1.350±0.010†)	(0.890±0.010†)	–	–
Sequential-VAE	(1.390±0.070†)	(0.920±0.090†)	–	–
GRU-D	(1.160±0.050†)	(0.790±0.060†)	–	–
GRU-ODE	0.839±0.030	0.479±0.044	0.876±0.589	0.365±0.012
	(0.830±0.040†)	(0.480±0.010†)	(0.748±0.045‡)	(0.379±0.005)
Neural-Flows	0.866±0.097	0.479±0.045	0.796±0.053	0.374±0.017
	(0.781±0.041‡)	(0.499±0.004‡)	(0.734±0.054‡)	(0.364±0.008‡)
GraFITi+	0.657±0.040	**0.401±0.028**	0.351±0.045	**0.233±0.005**
	–	(0.396±0.030♯)	–	(0.225±0.001♯)
ProFITi (ours)	**0.511±0.068**	0.474±0.049	**-0.762±0.119**	0.251±0.001

We run the experiment with the same protocol mentioned in the baseline papers. We see that ProFITi outperforms baseline models again. The gains provided by ProFITi in MNL is less pronounced than in NJNL as the model is designed and trained to learn joint distributions. Whereas for MSE, GraFITi that is trained for Gaussian MNL performs better than ProFITi. This is expected because Gaussian MNL has an MSE component in it and a model specifically trained for a metric tends to perform better than others for that metric. However, for (IMTS) probabilistic forecasting models the primary metric of interest is (marginal or normalized joint) negative log likelihood, where ProFITi performs the best (also see Section 7.1).

7.3 Ablation studies: Varying model components

We show the impact of different ProFITi components using Physionet2012 dataset. We see that the Shiesh activation function provides a significant improvement as it can help learning non-Gaussian distributions (compare ProFITi and ProFITi-Shiesh). Similarly, learning joint distributions (ProFITi) provides better NJNL compared to ProFITi-SITA. Learning only Gaussian marginal distributions (ProFITi-SITA-Shiesh) performs significantly worse than ProFITi. Using PReLU instead of Shiesh (ProFITi-Shiesh+PReLU) deteriorates the performance of ProFITi. Using Leaky-ReLU leads to very small Jacobians and also vanishing gradient problem, hence we avoid showing it. We see that A^{iTrans} (ProFITi-A^{tri} + A^{iTrans}) perform bad as it can learn only positive covariances. Finally, we see that ProFITi with either A^{reg} or A^{tri} performs comparably, however, A^{reg} has scalability problems as computing the determinant of the full attention matrix has computational complexity $\mathcal{O}(K^3)$, while for the triangular attention matrix only $\mathcal{O}(K)$. Also, it performs worse with increasing forecast lengths (see Section G.2).

Table 5: Varying model components. Shown is NJNL. ProFITi-A+B indicates component A is removed and B is added.

Model	Physionet2012
ProFITi	-0.766±0.038
ProFITi-SITA	-0.470±0.017
ProFITi-Shiesh	0.285±0.061
ProFITi-SITA-Shiesh	0.372±0.021
ProFITi-Shiesh+PReLU	0.384±0.060
ProFITi-A^{tri}+A^{iTrans}	-0.199±0.141
ProFITi-A^{tri}+A^{reg}	-0.778±0.016

Conclusions

In this work, we propose a novel model ProFITi for the probabilistic forecasting of irregularly sampled multivariate time series with missing values using conditioning normalizing flows. ProFITi is a permutation invariant normalizing flow model applied to learn conditional permutation invariant structured distributions. We propose two novel model components, sorted invertible triangular self attention and Shiesh activation function in order to learn any random target distribution. Our experiments on 3 IMTS datasets demonstrate that ProFITi provides better uncertainty than a range of probabilistic IMTS forecasting baselines.

REFERENCES

Govinda Anantha Padmanabha and Nicholas Zabaras. Solving inverse problems using conditional invertible neural networks. *Journal of Computational Physics*, 433:110194, May 2021. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110194. URL https://www.sciencedirect.com/science/article/pii/S0021999121000899.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 573–582. PMLR, May 2019. URL https://proceedings.mlr.press/v97/behrmann19a.html.

Marin Biloš and Stephan Günnemann. Normalizing flows for permutation invariant densities. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 957–967. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/bilos21a.html.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural odes. *Advances in Neural Information Processing Systems*, 34:21325–21337, 2021.

Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, 31, 2018.

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

Ricky T. Q. Chen, Jens Behrmann, David K Duvenaud, and Joern-Henrik Jacobsen. Residual flows for invertible generative modeling. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5d0d5594d24f0f955548f0fc0ff83d10-Paper.pdf.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for multivariate time series analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2995–3007. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1f47cef5e38c952f94c5d61726027439-Paper.pdf.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32, 2019.

Ruizhi Deng, Bo Chang, Marcus A Brubaker, Greg Mori, and Andreas Lehrmann. Modeling continuous stochastic processes with dynamic normalizing flows. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7805–7815. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/58c54802a9fb9526cd0923353a34a7ae-Paper.pdf.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

Robert Dürichen, Marco A. F. Pimentel, Lei Clifton, Achim Schweikard, and David A. Clifton. Multitask gaussian processes for multivariate physiological time-series analysis. *IEEE Transactions on Biomedical Engineering*, 62(1):314–322, 2015. doi: 10.1109/TBME.2014.2351376.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, pp. 7, 2019.

A Johnson, L Bulgarelli, T Pollard, S Horng, and LA Celi. Mark. *R. MIMIC-IV (version 1.0). PhysioNet*, 2021.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

AmirEhsan Khorashadizadeh, Konik Kothari, Leonardo Salsi, Ali Aghababaei Harandi, Maarten de Hoop, and Ivan Dokmanić. Conditional injective flows for bayesian imaging. *IEEE Transactions on Computational Imaging*, 9:224–237, 2023. doi: 10.1109/TCI.2023.3248949.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf.

Jonas Köhler, Leon Klein, and Frank Noe. Equivariant flows: Exact likelihood generative learning for symmetric densities. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5361–5370. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/kohler20a.html.

Rahul Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A conditional flow-based model for stochastic video generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rJgUfTEYvH.

Steven Cheng-Xian Li and Benjamin Marlin. A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 1812–1820, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Steven Cheng-Xian Li and Benjamin M Marlin. Classification of sparse and irregularly sampled time series with mixtures of expected gaussian kernels and random features. In *UAI*, pp. 484–493, 2015.

Yang Li, Haidong Yi, Christopher Bender, Siyuan Shan, and Junier B Oliva. Exchangeable neural ode for set modeling. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6936–6946. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4db73860ecb5533b5a6c710341d5bbec-Paper.pdf.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435.

Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs M Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=WiGQBFuVRv.

Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Bernd Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=N5hQI_RowVA.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 19388–19405. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/schirmer22a.html.

Randolf Scholz, Stefan Born, Nghia Duong-Trung, Mariano Nicolas Cruz-Bournazou, and Lars Schmidt-Thieme. Latent linear ODEs with neural kalman filtering for irregular time series forecasting, 2023. URL https://openreview.net/forum?id=a-bD9-0ycs0.

Satya Narayan Shukla and Benjamin Marlin. Heteroscedastic temporal variational autoencoder for irregularly sampled time series. In *International Conference on Learning Representations*, 2022.

Phillip Si, Volodymyr Kuleshov, and Allan Bishop. Autoregressive quantile flows for predictive uncertainty estimation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=z1-I6rOKv1S.

Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, pp. 245–248. IEEE, 2012.

Rhea Sanjay Sukthanker, Zhiwu Huang, Suryansh Kumar, Radu Timofte, and Luc Van Gool. Generative flows with invertible attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11234–11243, 2022.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

Brian L Trippe and Richard E Turner. Conditional density estimation with bayesian normalising flows, 2018.

Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. In Amir Globerson and Ricardo Silva (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 393–402. AUAI Press, 2018. URL http://auai.org/uai2018/proceedings/papers/156.pdf.

Cédric Villani. *Optimal Transport*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71049-3. doi: 10.1007/978-3-540-71050-9. URL http://link.springer.com/10.1007/978-3-540-71050-9.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/2a084e55c87b1ebcdaad1f62fdbbac8e-Paper.pdf.

Antoine Wehenkel and Gilles Louppe. Graphical normalizing flows. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 37–45. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/wehenkel21a.html.

Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows, 2019.

Vijaya Krishna Yalavarthi, Kiran Madusudanan, Randolf Scholz, Nourhan Ahmed, Johannes Burchert, Shayan Javed, Stefan Born, and Lars Schmidt-Thieme. Forecasting irregularly sampled time series using graphs. *arXiv preprint arXiv:2305.12932*, 2023.

Jiajun Zha, Yiran Zhong, Jing Zhang, Richard Hartley, and Liang Zheng. Invertible attention. *CoRR*, abs/2106.09003, 2021. URL https://arxiv.org/abs/2106.09003.

Table 6: Statistics of the datasets used our experiments. Sparsity means the percentage of missing observations in the time series

Name	#Samples	#Chann.	Max. len.	Max. Obs.	Sparsity
Physionet'12	12,000	37	48	520	85.7%
MIMIC-III	21,000	96	96	710	94.2%
MIMIC-IV	18,000	102	710	1340	97.8%

A Dataset Details

Three datasets are used for evaluating the proposed model. Basic statistics of the datasets is provided in Table 6.

Physionet2012 Silva et al. (2012) encompasses the medical records of 12,000 patients who were hospitalized in the ICU. During the initial 48 hours of their admission, 37 vital signs were measured. We follow the protocol used in previous studies Che et al. (2018); Cao et al. (2018); Tashiro et al. (2021); Yalavarthi et al. (2023). After pre-processing, dataset consists of hourly observations making a total of up to 48 observations in each series.

MIMIC-III Johnson et al. (2016) constitutes a medical dataset containing data from ICU patients admitted to Beth Israeli Hospital. 96 different variables from a cohort of 18,000 patients were observed over an approximately 48-hour period. Following the preprocessing procedures outlined in (Biloš et al., 2021; De Brouwer et al., 2019; Yalavarthi et al., 2023), we rounded the observations to 30-minute intervals.

MIMIC-IV Johnson et al. (2021) is an extension of the MIMIC-III database, incorporating data from around 18,000 patients admitted to the ICU at a tertiary academic medical center in Boston. Here, 102 variables are monitored. We followed the preprocessing steps of (Biloš et al., 2021; Yalavarthi et al., 2023) and rounded the observations to 1 minute interval.

B Implementing CNF+

We detail the CNF+ model that is used for comparison in Section 7.1. First, to the best of our knowledge, there exists no continuous normalizing flow that can be applied directly to the current problem setup of predicting conditional density of sequences with variable lengths. Hence, we inspire from the CNF proposed by (Biloš & Günnemann, 2021), we implement CNF+ that can be applied for our case. We concatenate the conditioning inputs x and answers y. Used canonical dot product attention as the vector filed g and $[x\|y]$ as $v(0)$ in eq. 4. The output of the continuous flow $v(1)$ is considered z.

C Invertibility of A^{reg}

We prove that A^{reg} presented in Section 4 is invertible.

Lemma 1. *For any $K \times K$ matrix A and $\epsilon > 0$, the matrix $\mathbb{I}_K + \frac{1}{\|A\|_2+\epsilon}A$ is invertible. Here, $\|A\|_2 \coloneqq \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|}$ denotes the spectral norm.*

Proof. Assume it was not the case. Then there exists a non-zero vector x such that $(\mathbb{I}_K + \frac{1}{\|A\|_2+\epsilon}A)x = 0$. But then $(\|A\|_2 + \epsilon)x = -Ax$, and taking the norm on both sides and rearranging yields $\|A\|_2 \geq \frac{\|Ax\|_2}{\|x\|_2} = \|A\|_2 + \epsilon > \|A\|_2$, contradiction! Hence the lemma. □

D UNCONSTRAINED MONOTONIC NEURAL NETWORKS ARE JUST CONTINUOUS NORMALIZING FLOWS

Any unconstrained monotonic neural network (Wehenkel & Louppe, 2019) can equivalently be written as a standard continuous normalizing flow. To make our deduction of the Shiesh activation function in section slightly more streamlined, we therefore have presented unconstrained monotonic neural network as continuous normalizing flows from the beginning.

Lemma 2. *Any UMNN function a^{UMNN} defined by*

$$a^{UMNN}(u) := \int_0^u f(\tau)d\tau + b$$

with a positive function f can be represented as a continuous normalizing flow for a suitable scalar field g:

$$a^{CNF}(u) := v(1) \quad \textit{with } v : \mathbb{R} \to \mathbb{R} \textit{ being the solution of } \frac{\partial v}{\partial \tau} = g(\tau, v(\tau)), \quad v(0) := u$$

Proof. Let $a^{\text{UMNN}} : \mathbb{R} \to \mathbb{R}$, $u \mapsto a^{\text{UMNN}}(u)$ be a UMNN function. The continuous normalizing flow to be constructed must connect each $(0, u)$ by a flowline to $(1, a^{\text{UMNN}}(u))$ in the product space $[0, 1] \times \mathbb{R}$ (see Figure 4). The easiest way to do this is via a line segment, namely the flowline

$$\phi_t(u) = (0, u) + t(1, a^{\text{UMNN}}(u) - u) \ , \ t \in [0, 1] \ .$$

These lines do not intersect, as for all $t \in [0, 1]$ the second coordinate is strictly monotounously increasing with respect to u: $\frac{\partial \phi_t(u)}{\partial u} = (0, (1 - t) + a^{\text{UMNN}'}(u))$, $(1 - t) + a^{\text{UMNN}'}(u) > 0$ (by UMNN).

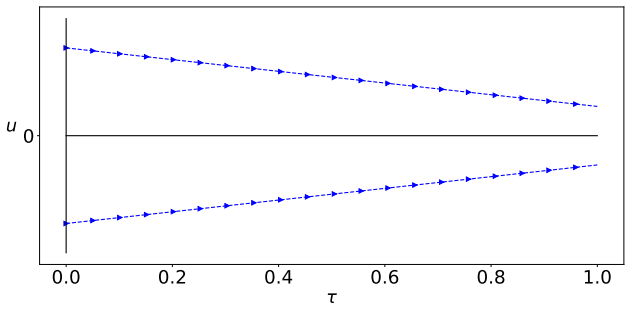


Figure 4: Demonstrating UMNN flowlines

We now have to write the curves $\phi_t(u)$ as integral curves of a time dependent vector field $g(t, u)$ on $[0, 1] \times \mathbb{R}$, or rather the second component of $\phi_t(u)$, namely $a_t(u) = u(1 - t) + a^{\text{UMNN}}(u)t$ as the solution of a differential equation

$$\frac{\partial a_t(u)}{\partial t} = g(t, a_t(u)) = g(\phi_t(u)) \ .$$

Now $\frac{\partial a_t(u)}{\partial t} = a^{\text{UMNN}}(u) - u$, so we have to set $g(\phi_t(u)) := a^{\text{UMNN}}(u) - u$ for any $t \in [0, 1]$ and $u \in \mathbb{R}$. We can explicitly write $g(t, u) = a^{\text{UMNN}}(a_t^{-1}(u)) - a_t^{-1}(u)$. The inverse of the continuously differentiable function a_t with positive derivative exists and is continuously differentiable. □

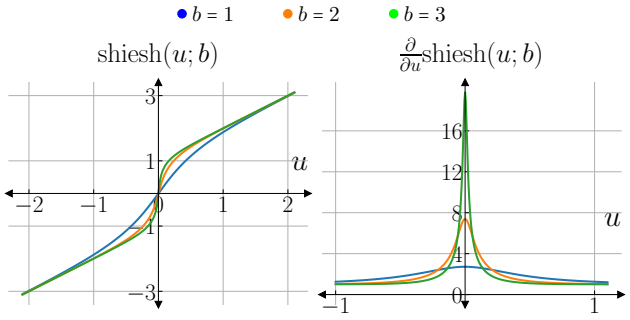


Figure 5: Demonstration of Shiesh activation function with varying b.

E Shiesh ACTIVATION FUNCTION

E.1 SOLVING ODE

The differential equation $\frac{dv(\tau)}{d\tau} = \tanh(bv(\tau)), \quad v(0) := u$ can be solved by separation of variables. However, we can also proceed as follows by multiplying the equation with $b\cosh(b \cdot v(\tau))$:

$$
\begin{aligned}
& b\cosh(bv(\tau))\frac{dv(\tau)}{d\tau} = b\sinh(bv(\tau)) \\
\Leftrightarrow \quad & \frac{d\sinh(bv(\tau))}{d\tau} = b\sinh(bv(\tau)) \\
\Leftrightarrow \quad & \sinh(bv(\tau)) = Ce^{b\tau} \quad \text{for some } C \\
\Leftrightarrow \quad & v(\tau) = \frac{1}{b}\sinh^{-1}(Ce^{b\tau}) \quad \text{for some } C\,.
\end{aligned}
$$

The initial condition yields $C = \sinh(bu)$

E.2 INVERTIBILITY OF Shiesh

A function $F : \mathbb{R} \to \mathbb{R}$ is invertible if it is strictly monotonically increasing.

Theorem 1. *Function Shiesh*$(u; b) = \frac{1}{b}\sinh^{-1}(e^b \sinh(b \cdot u))$ *is strictly monotonically increasing for* $u \in \mathbb{R}$.

Proof. A function is strictly monotonically increasing if its first derivate is always positive. From eq. 9, $\frac{\partial}{\partial u}\text{Shiesh}(u; b) := \frac{e^b \cosh(b \cdot u)}{\sqrt{1+(e^{b\cdot\tau}\sinh(b\cdot u))^2}}$. We known that $e^{b\cdot\tau}$ and $\cosh(u)$ are always positive hence $\frac{\partial}{\partial u}\text{Shiesh}(u; b)$ is always positive. □

E.3 IMPLEMENTATION DETAILS

Implementing Shiesh on the entire $\mathbb{R}$ will have numerical overflow. Hence, we implement it in piece-wise manner. In this work, we are interested in $b > 0$ and show all the derivations for it.

With $\sinh(x) = \frac{e^x - e^{-x}}{2}$ and $\sinh^{-1}(x) = \log(x + \sqrt{1 + x^2})$ Shiesh can be rewritten as follows:

$$
\begin{aligned}
\text{Shiesh}(u; b) &:= \frac{1}{b}\sinh^{-1}\Big(\exp(b) \cdot \sinh(b \cdot u)\Big) \\
&= \frac{1}{b}\log\Big(\exp(b) \cdot \sinh(b \cdot u) + \sqrt{1 + \Big(\exp(b) \cdot \sinh(b \cdot u)\Big)^2}\Big) \\
&= \frac{1}{b}\log\left(\Big(\exp(b) \cdot \frac{\exp(b \cdot u) - \exp(-b \cdot u)}{2}\Big) + \sqrt{1 + \Big(\exp(b) \cdot \frac{\exp(b \cdot u) - \exp(-b \cdot u)}{2}\Big)^2}\right)
\end{aligned}
$$

When $u \gg 0$, Shiesh can be approximated to the following:

$$\begin{aligned}
\text{Shiesh}(u;b) &\approx \frac{1}{b}\log\left(\exp(b)\cdot\left(\frac{\exp(b\cdot u)}{2}\right)+\sqrt{1+\left(\exp(b)\cdot\frac{\exp(b\cdot u)}{2}\right)^2}\right), \qquad \exp(-b\cdot u)\to 0\\
&\approx \frac{1}{b}\log\left(\exp(b)\cdot\frac{\exp(b\cdot u)}{2}+\exp(b)\cdot\frac{\exp(b\cdot u)}{2}\right), \qquad \sqrt{1+u^2}\approx u \quad \text{for} \quad u\gg 0\\
&= \frac{1}{b}\log\left(\exp(b)\cdot\exp(b\cdot u)\right)\\
&= \frac{1}{b}\log\left(\exp(b)\right)+\frac{1}{b}\log\left(\exp(b\cdot u)\right)\\
&= 1+u
\end{aligned}$$

Now for $u \ll 0$, we know that $\sinh^{-1}(u)$ and $\sinh(u)$ are odd functions meaning

$$\sinh^{-1}(-u) = -\sinh^{-1}(u) \tag{14}$$
$$\sinh(-u) = -\sinh(u) \tag{15}$$

Also, we know that composition of two odd functions is an odd function making Shiesh an odd function. Now,

$$\begin{aligned}
&\text{Shiesh}(u;b)\approx u+1 && \text{for} \quad u>>0\\
\implies &\text{Shiesh}(u;b)\approx -(-u+1) && \text{for} \quad u<<0
\end{aligned}$$

Hence, to avoid numerical overflow in implementing Shiesh, we apply it in piece-wise manner as follows:

$$\text{Shiesh}(u;b) = \begin{cases}\frac{1}{b}\sinh^{-1}(\exp(b)\sinh(b\cdot u)) & \text{if} \quad |x|\le 5\\ u+1\cdot\text{sign}(u) & \text{else}\end{cases}$$

Similarly, its partial derivative is implemented using:

$$\frac{\partial}{\partial u}\text{Shiesh}(u;b) = \begin{cases}\frac{e^b\cosh(b\cdot u)}{\sqrt{1+\left(e^b\sinh(b\cdot u)\right)^2}} & \text{if} \quad |x|\le 5\\ 1 & \text{else}\end{cases} \tag{16}$$

E.4 Bounds of the derivatives

Assume $\mathbf{D}\text{Shiesh}(u;b) = \frac{\partial}{\partial u}\text{Shiesh}(u;b)$ and $b > 0$. For larger values of u, from eq. 16, $\mathbf{D}\text{Shiesh}(u;b) \approx 1$. Now we show that for the values $u \in [-5,5]$ the maximum value for $\mathbf{D}\text{Shiesh}(u;b)$. For this we compute $\mathbf{D}^2\text{Shiesh}(u;b)$:

$$\begin{aligned}
\mathbf{D}^2\text{Shiesh}(u;b) &:= \frac{be^b\sinh(bu)\left(e^{2b}\sinh^2(bu)-e^{2b}\cosh^2(bu)+1\right)}{\left(e^{2b}\sinh^2(bu)+1\right)^{3/2}}\\
&:= \frac{be^b\sinh(bu)(1-e^{2b})}{\left(e^{2b}\sinh^2(bu)+1\right)^{3/2}}
\end{aligned}$$

In order to compute the maximum of the function $\mathbf{D}\text{Shiesh}(u;b)$, we equate $\mathbf{D}^2\text{Shiesh}(u;b)$ to zero:

$$\begin{aligned}
&be^b\sinh(bu)(1-e^{2b}) = 0 \quad \left(\left(e^{2b}\sinh^2(bu)+1\right)^{3/2}>0\right)\\
&\implies \sinh(bu) = 0\\
&\implies u = 0
\end{aligned}$$

```python
class shiesh(nn.Module):
    threshold: Tensor
    slope: Tensor

    def __init__(self) -> None:
        super().__init__()
        self.register_buffer("threshold", 5)
        self.register_buffer("slope", torch.exp(1.))

    def shiesh_(self, x: Tensor) -> Tensor:
        return torch.archsinh(torch.exp(1)*torch.sinh(x))

    def Dshiesh(self, x: Tensor) -> Tensor:
        return torch.exp(1)*torch.cosh(x)/(1 + (torch.exp(1)*torch.sinh(x))**2)**0.5

    def forward(self, x: Tensor) -> (Tensor, Tensor):
        mask = x.abs() <= self.threshold
        x_out =  torch.where(mask, self.shiesh_(x), x + torch.sign(x))
        ldj = torch.log(torch.where(mask, self.Dshiesh(x), x + torch.sign(x)))
        return x_out, ldj
```

(a) Pytorch implementation of Shiesh.

```python
class shiesh_inv(nn.Module):
    threshold: Tensor
    slope: Tensor

    def __init__(self) -> None:
        super().__init__()
        self.register_buffer("threshold", 5)
        self.register_buffer("slope", torch.exp(1.))

    def shiesh_inv_(self, x: Tensor) -> Tensor:
        return torch.archsinh(torch.exp(-1)*torch.sinh(x))

    def Dshiesh_inv(self, x: Tensor) -> Tensor:
        return torch.exp(-1)*torch.cosh(x)/(1 + (torch.exp(-1)*torch.sinh(x))**2)**0.5

    def forward(self, x: Tensor) -> (Tensor, Tensor):
        mask = x.abs() <= self.threshold
        x_out =  torch.where(mask, self.shiesh_inv_(x), x - torch.sign(x))
        ldj = torch.log(torch.where(mask, self.Dshiesh_inv(x), x + torch.sign(x)))
        return x_out, ldj
```

(b) Pytorch implementation of Shiesh^{-1} .

Figure 6: Implementation of Shiesh and its inverse in Pytorch.

Now, we compute $\mathbf{D}^3\text{Shiesh}(u;b)$ for $u = 0$. $\mathbf{D}^3\text{Shiesh}(u;b)$ can be given as:

$$\mathbf{D}^3\text{Shiesh}(u;b) = -\frac{b^2e^b(2e^{2b}\sinh^2(bu)-1)\cosh(bx)(e^{2b}\sinh^2(bu)-e^{2b}\cosh^2(bu)+1)}{(e^{2b}\sinh^2(bu)+1)^{5/2}}$$

Substituting $u = 0$, we get

$$\begin{aligned}\mathbf{D}^3\text{Shiesh}(0;b) &= -\frac{b^2e^b(2e^{2b}\cdot 0-1)\cdot 1\cdot(e^{2b}\cdot 0-e^{2b}\cdot 1+1)}{(e^{2b}\cdot 0+1)^{5/2}} \\ &= b^2e^b(1-e^{2b}) \ < 0 \qquad (b>0)\end{aligned}$$

Hence, the bounds for the $\mathbf{D}\text{Shiesh}(u;b)$ is $\{1, e^b\}$.

F Creating toy example for conditional heteroscedastic permutation equivariant distributions: Figure 1

Here, we show how to generate the toy example used in the Section 1. It is a mixture of two bivariate Gaussian distributions. We first generate the conditioning variables $x_k \sim \mathbb{N}(0,1), k = 1:2$ (Eq. 17). Then, we use the generated x to create a covariance matrix Σ (Eq. 18). Now, we draw the samples using the mixture of Gaussians as in Eq 19. We allow large gap between two Gaussians so

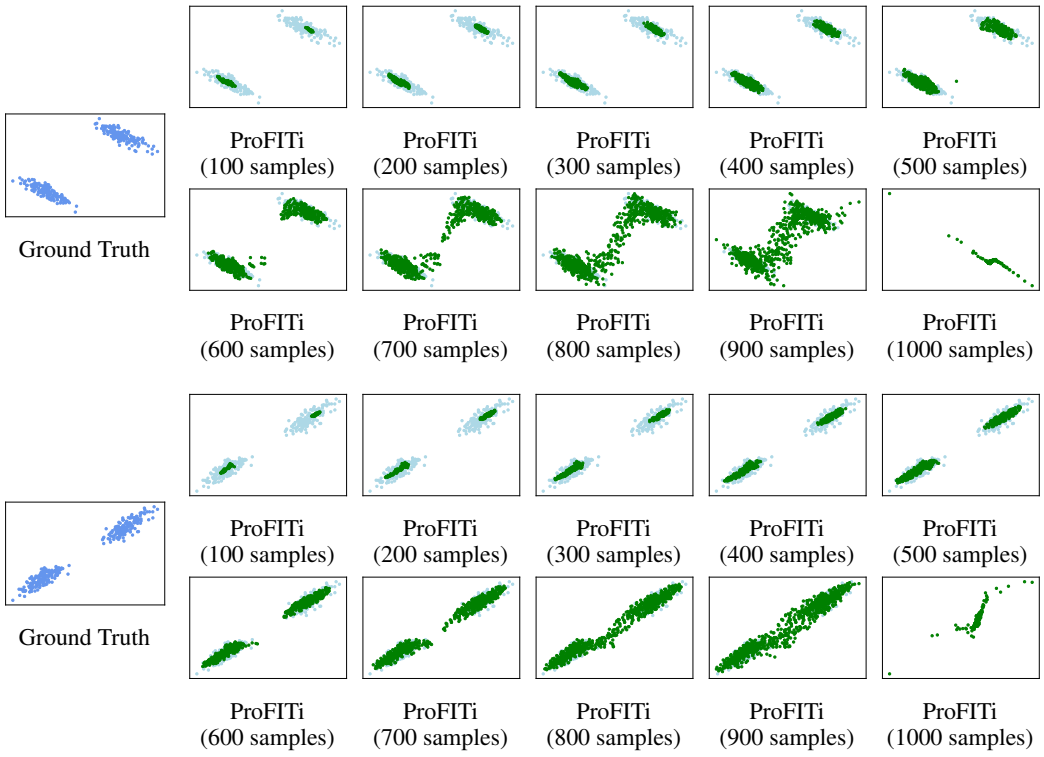


Figure 7: Demonstrating the distributions generated by ProFITi. MC sampling of 1000, sorted them with increasing likelihood. With increase in samples after sorting, the distribution deviates from the true distribution. For the images showing distributions of ProFITi, Ground Truth distribution is shown in the background.

that the plots can look separable.

$$x_k \sim \mathcal{N}(0,1), \quad k \in 1{:}2, \quad x^{\text{com}} := () \tag{17}$$

$$\Sigma(x) := \begin{pmatrix} 1+|x_1| & \operatorname{sgn}(x_1x_2)\sqrt{(1+|x_1|)(1+|x_2|)-1} \\ \operatorname{sgn}(x_1x_2)\sqrt{(1+|x_1|)(1+|x_2|)-1} & 1+|x_2| \end{pmatrix} \tag{18}$$

$$y_{1:2} \sim \mathcal{N}\left(\begin{bmatrix}5+x_{1,1}\\5+x_{2,1}\end{bmatrix}), \Sigma(x_{.,2})\right) + \mathcal{N}\left(\begin{bmatrix}-5+x_{1,1}\\-5+x_{2,1}\end{bmatrix}), \Sigma(x_{.,2})\right) \tag{19}$$

For GPR, we implemented (Dürichen et al., 2015), whereas for Generalized Linear Model, we simply pass the (x_1, x_2) to a single layer feed forward neural network and predicted mean and standard deviation of a normal distribution.

In Figure 7, we demonstrate the density generated by ProFITi. We randomly generated 1000 samples and sorted them according to their likelihoods. Then, we plot the density of those sorted samples in the increase order. As expected with all the samples (ProFITi (1000 samples)), samples with least likelihood will fall far outside the true distribution.

G Additional experiments

G.1 Experiment on varying the order of the channels

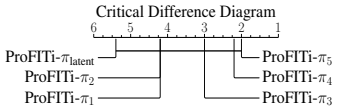


Figure 8: Statistical test on the results of various channel orders for ProFITi.

In ProFITi, we fix the order of channels to make it equivariant. In Figure 8, through critical difference diagram, we demonstrate that changing the permutation used to fix the channel order does not provide statistically significant difference in the results. ProFITi$-\pi_{1:5}$ indicate ProFITi with 5 different pre-fixed permutations on channels while time points are left in causal order. The order in which we sort channels and time points is a hyperparameter. To avoid this hyperparamerter and even allow different sorting criteria for different instances, one can parametrize P_π as a function of $X_{1:|X|-1,.}$ (**learned sorted triangular invertible self attention**). ProFITi$-\pi_{\text{latent}}$ indicate ProFITi where the permutation of all the observations (including channels and time points) are set on the latent embedding. Specifically, we pass h through an MLP and selected the permutation by sorting its output. Significant difference in results is not observed because the ordering in lower triangular matrix can be seen as a Bayesian network, and the graph with the triangular matrix as adjacency is a full directed graph, and all of them induce the same factorization. Also, a triangular linear map $z \mapsto \mathrm{L}z$ to a distribution can describe any covariance matrix Σ via a Cholesky decomposition $\Sigma = \mathrm{L}^T\mathrm{L}$, as $\rho(\mathrm{L}z) = ce^{-\frac{1}{2}z^T\mathrm{L}^T\mathrm{L}z}$.

G.2 Scalability Experiment

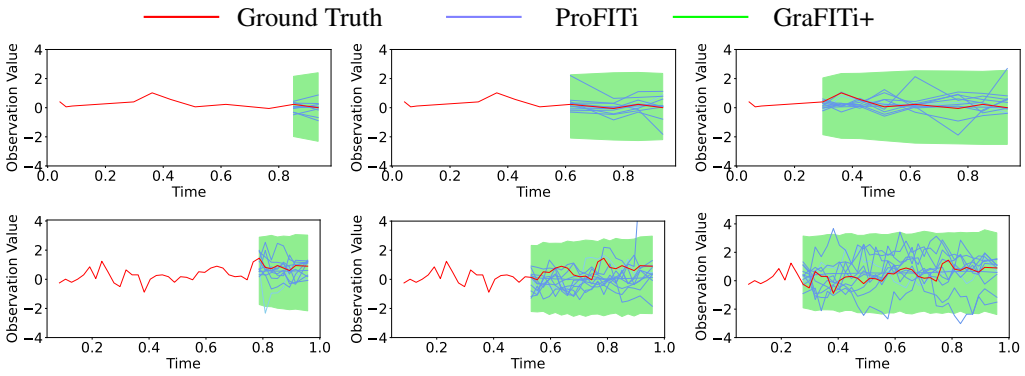
In Table 7, we compare ProFITi with two next best models, GraFITi+ and Neural Flows. Our evaluation involves varying the observation and forecast horizons on the Physionet'12 dataset. Furthermore, we also compare with ProFITi-A^{tri}+A^{reg}, wherein the triangular attention mechanism in ProFITi is replaced with a regularized attention mechanism.

ProFITi exhibits superior performance compared to both Neural Flows and GraFITi+, demonstrating a significant advantage. We notice that when we substitute A^{tri} with A^{reg}; this change leads to a degradation in performance as the forecast sequence length increases. Also, note that the run time for computing A^{reg} and its determinant is an order of magnitude larger than that of A^{tri}. This is because, it requires $\mathcal{O}(K^3)$ complexity to compute spectral radius $\sigma(A)$ and determinant of A^{reg}, whereas computing determinant of A^{tri} requires $\mathcal{O}(K)$ complexity.

Additionally, we see that as the sequence length increases, there is a corresponding increase in the variance of the NJNL. This phenomenon can be attributed to the escalating number of target values (K), which increases with longer sequences. Predicting the joint distribution over a larger set of target values can introduce noise into the results, thereby amplifying the variance in the outcomes. Whereas for the GraFIT+ and Neural Flows it is not the case as they predict only marginal distributions. Further, as expected the NJNL of all the models decrease with increase in sequence lengths as it is difficult to learn longer horizons compared to short horizons of the forecast.

Table 7: Varying observation and forecast horizons of Physionet'12 dataset

	obs/forc : 36/12hrs			obs/forc : 24/24hrs			obs/forc : 12/36hrs		
	NJNL	run time (s)		NJNL	run time (s)		NJNL	run time (s)	
		epoch	A		epoch	A		epoch	A
Neural Flows	0.709±0.483	109.6	-	1.097±0.044	46.6	-	1.436±0.187	45.5	-
GraFITi+	0.522±0.015	42.9	-	0.594±0.009	43.1	-	0.723±0.004	37.5	-
ProFITi	-0.768±0.041	64.8	3.3	-0.355±0.243	66.2	5.2	-0.291±0.415	82.1	8.6
ProFITi-A^{tri}+A^{reg}	-0.196±0.096	89.9	7.1	0.085±0.209	142.1	30.1	0.092±0.168	245.8	73.1



(a) Obs/forc: 36/12hrs (b) Obs/forc: 24/24hrs (c) Obs/forc: 12/36hrs

Figure 9: Demonstrating (10) trajectories generated using ProFITi for Physionet'12 dataset.

In Figure 9, we show the qualitative performance of ProFITi. We compare the trajectories predicted by ProFITi by random sampling of z with the distribution predicted by the GraFITi+ (next best model).

H HYPERPARAMETERS SEARCHED

Following the original works of the baseline models, we search the following hyperparameters:

HETVAE (Shukla & Marlin, 2022) :

- Latent Dimension: {8, 16, 32, 64, 128}
- Width : {128,256,512}
- # Reference Points: {4, 8, 16, 32}
- # Encoder Heads: {1, 2, 4}
- MSE Weight: {1, 5, 10}
- Time Embed. Size: {16, 32, 64, 128}
- Reconstruction Hidden Size: {16, 32, 64, 128}

GRU-ODE-Bayes (De Brouwer et al., 2019) :

- solver: {euler, dopri5}
- # Hidden Layers: {3}
- Hidden Dim.: {64}

Neural Flows (Biloš et al., 2021) :

- Flow Layers: {1, 4}

- # Hidden Layers: {2}
- Hidden Dim.: {64}

CRU (Schirmer et al., 2022) :

- # Basis: {10, 20}
- Bandwidth: {3, 10}
- lsd: {10, 20, 30}

CNF+ :

- # Attention layers: {1,2,3,4}
- # Projection matrix dimension for attention: {32,64,128,256}

GraFITi+ (Yalavarthi et al., 2023) :

- # layers: {2, 3, 4}
- # MAB heads: {1, 2, 4}
- Latent Dim.: {32, 64, 128}

ProFITi (Ours) :

- # Flow layers: {8, 9, 10}
- ϵ: {0.1}
- Latent Dim.: {32, 64, 128, 256}