Applied Soft Computing Journal 186 (2026) 114039

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Embracing data irregularities in multivariate time series☆

Marcel Barros[a,*], Lucas P. Fontenele[a], Mariana S. Silva[a], Thiago Rissi[a], Eduardo Cabrera[a], Eduardo A. Tannuri[a], Edson S. Gomi[a], Rodrigo A. Barreira[b], Anna H. Reali Costa[a]

[a] Escola Politécnica da Universidade de São Paulo, São Paulo, SP, Brazil
[b] Petróleo Brasileiro S.A., Rio de Janeiro, RJ, Brazil

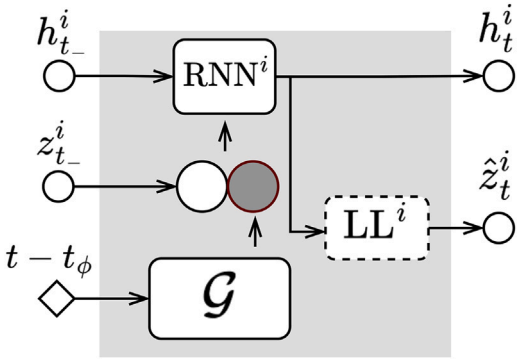
HIGHLIGHTS

- A novel architecture is proposed to address irregularities in multivariate time series (MTS), including varying sampling rates, missing data, and time misalignments.
- The model combines time-informed recurrent neural networks (RNNs) with graph neural networks (GNNs) for generalized temporal modeling.
- A modified training method enables the model to generalize temporal patterns and generate forecasts at unseen frequencies.
- Validation on real-world data from Floating Production, Storage, and Offloading units (FPSOs) demonstrates superior predictive accuracy and reduced preprocessing requirements.

ARTICLE INFO

Keywords:
Multivariate time series
Machine learning
Recurrent neural networks

GRAPHICAL ABSTRACT



GATE block setup. The transformation performed by the sequence model (here, an RNN) incorporates $z^i_{t_-}$ into $h^i_{t_-}$ and translates the hidden representation forward to time t. The resulting state h^i_t is projected by a linear layer (LL) to produce the prediction $\hat{z}^i_t$.

ABSTRACT

Data collection in many engineering fields involves multivariate time series gathered from a collection of sensors that operate independently of each other. These sensors often display various irregularities, such as different sampling rates and missing data. To manage these issues, complex preprocessing mechanisms are required, which become coupled with any statistical model trained with the transformed data. Modeling the motion of floating platforms anchored on seabeds from measurements acquired from sensors is a typical example. We propose and analyze a model in which each sensor is encoded using an independent time-informed recurrent neural network, information is propagated in a common latent space by a graph neural network, and a modified training method is used to induce temporal generalization of the model. Our method can generate forecasts at unseen frequencies, which provides empirical evidence that the model learns an implicit representation of the system's time derivatives and is able to flexibly integrate the signal over the time domain.

☆ We gratefully acknowledge the support from ANP/PETROBRAS, Brazil (project N. 21721-6), CNPq (grant numbers 312360/2023-1, 310127/2020-3) and CAPES (Finance Code 001).

* Corresponding author.
Email address: marcel.barros@usp.br (M. Barros).

https://doi.org/10.1016/j.asoc.2025.114039
Received 13 January 2025; Received in revised form 20 September 2025; Accepted 1 October 2025
Available online 9 October 2025

1. Introduction

Despite the fact that multivariate time series (MTS) research has been ongoing for more than four decades, it is still not as thoroughly investigated as univariate time series. This discrepancy arises partly from the unique features of MTS, which diverge from traditional statistical principles. Although classical approaches usually depend on random sampling and the assumption of independence, MTS are typically marked by strong correlations [1].

However, the field has garnered attention from Machine Learning (ML) researchers, motivated by recent achievements in token sequence modeling, particularly in tasks related to Natural Language Processing (NLP). Models like the Transformer [2] have generated groundbreaking results in tasks such as question answering and text classification, thereby prompting the question: Can such successes be replicated in the context of time series?

For most applications, ML algorithms perform poorly compared to traditional statistical methods for univariate time series [3]. One of the main hurdles is that deep learning models, largely responsible for the ML revolution in NLP, are particularly susceptible to overfitting in univariate scenarios. However, multivariate problems exhibiting complex interactions are not easily modeled by classical approaches. This makes it an attractive area for ML research.

This work introduces a cost-effective encoder–decoder architecture based on recurrent and graph neural networks (RNNs and GNNs), coupled with a time representation capable of forecasting MTS. Moreover, this proposed model can process data from sensors with differing sampling rates and missing data profiles without resorting to data imputation techniques, which are often unfit for highly irregular MTS obtained from uncoordinated sensors. Our dataset consists of measured motion data from Floating Production, Storage, and Offloading units (FPSOs)—floating sea platforms used for oil extraction. Moored to the seafloor, FPSOs display intricate oscillatory behavior influenced by environmental factors such as wind, currents, and waves. Considering the successes obtained in this complex real-world scenario, we argue that the proposed architecture can be successful in a wide range of applications.

The key contributions of this work are fourfold.

- We propose a definition for MTS and its associated forecasting task that accounts for irregularities.
- Based on this definition, we propose a comprehensive end-to-end system capable of modeling FPSO hydrodynamic behavior at low computational costs and without the need for elaborate data preprocessing routines.
- We demonstrate that the combination of time encoding and masked training improves the modeling of complex oscillatory dynamics represented as time series, especially when large missing data windows are present.
- We provide evidence that RNNs trained with our method exhibit temporal generalization, being able to generate target sequences with sampling frequencies unseen during training.

In light of the above, this article is organized as follows. Section 2 presents the foundational concepts that underlie this work and provides a synopsis of how data irregularities can affect common ML architectures. Section 3 outlines the proposed architecture, while Section 4 offers a more detailed description of the dataset and experimental setup. Sections 5 and 6, respectively, present the experimental results and offer some conclusions, as well as directions for future work.

2. Background and definitions

A multivariate time series (MTS) $\mathcal{M}$ is often defined as a sequence $\mathbf{Z} = [z_1, \dots, z_L]$ with $z_t \in \mathbb{R}^K$, or equivalently as a matrix $\tilde{\mathbf{Z}} \in \mathbb{R}^{L\times K}$ where L is the number of observations and K the number of variables [1]. Most works assume a common, constant sampling rate, replacing the timestamp vector $\mathbf{t} = [t_1, \dots, t_L]$ by the trivial grid $\mathbf{t} = [1, \dots, L]$. When explicit timestamps are provided, they are treated as identical across variables, reducing any irregularity to missing values on this grid.

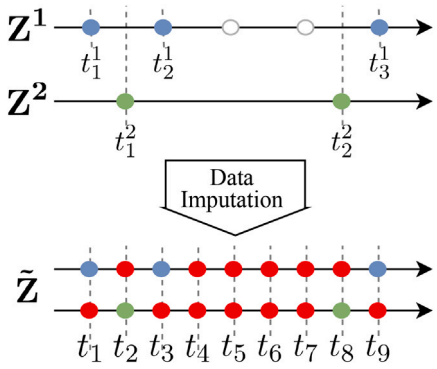


Fig. 1. Regularizing an irregular MTS may explode the number of inferred values. In this toy case the observed series has 5 points, yet its grid-aligned version requires 18. Red circles denote imputed entries.

To restore a regular grid, practitioners routinely impute missing data or supply a missing value mask. Classic imputation families include [4]: *mean imputation*, *linear interpolation*, *k-nearest neighbors*, *multiple imputation*, and *matrix factorization*. While this enables the direct use of RNNs, Transformers and MLPs on $\tilde{\mathbf{Z}}$ [5], the predictive model becomes tightly coupled to the chosen imputation strategy. Moreover, as Fig. 1 illustrates, combining misalignment, variable sampling rates, and sparsity can inflate the grid dramatically, making full regularization impractical.

An alternative is to let the model reason about missingness explicitly via a binary mask $\mathbf{M} \in \{0,1\}^{L\times K}$:

$$M_{t,k} = \begin{cases} 0, & Z_{t,k}\,\text{missing}, \\ 1, & Z_{t,k}\,\text{observed}. \end{cases}$$

The augmented input $[\mathbf{Z}, \mathbf{M}] \in \mathbb{R}^{L\times 2\ K}$ is then fed to the network, allowing it to learn directly from the available entries while being aware of the gaps [6,7].

2.1. Removing regularity assumptions

Irregularities in MTS data fall into four broad classes:

- **Irregular sampling** — uneven collection frequencies across sensors or time.
- **Missing data** — partial gaps in individual variables (distinct from full-sample irregularity).
- **Time offsets** — misalignments that quickly sparsify the implicit grid (cf. Fig. 1).
- **Heterogeneous formats** — variables may be images, audio, or text rather than simple scalars.

The early variants of RNN for continuous systems—Phased-LSTM [8], GRU-D [7], Neural ODE (NODE) [9]—already addressed uneven timestamps and introduced masks for missing entries, building on the continuous-time foundations of Ref. [10]. In NODEs, a continuous latent trajectory is modeled by $f : \mathcal{T} \to \mathcal{Y} \subset \mathbb{R}^K$, whose dynamics obey an ODE [11]. Real measurements, however, are discretized approximations; limited temporal resolution can register multiple events at the same recorded instant, as in particle-physics detectors [12]. Hence f need not be bijective in practice.

In a more significant relaxation step, each variable can be treated as an independent sequence or encoded as triplets (t, z, m), decoupling their sampling grids [13–15]. We adopt this per-variable view, which mirrors real sensor networks and highlights the complications of forcing misaligned data into a single regular grid (Fig. 1).

Definition 1 (Multivariate Time Series). An MTS is a pair of finite sets of sequences $\mathcal{M} = (\mathcal{S}, \mathcal{T}_S)$. The set $\mathcal{S} = \{\mathbf{Z^i}\}_{i=1}^{|S|}$ contains $|S|$ sequences $\mathbf{Z^i} = [z^i_{t_1}, z^i_{t_2}, \dots, z^i_{t_{L^i}}]$, where each measurement $z^i_t \in \mathcal{X}^i$ is associated with a timestamp $t \in \mathbb{R}$. The set $\mathcal{T}_S = \{\mathbf{T^i}\}_{i=1}^{|S|}$ contains all non-decreasing sequences of timestamps $\mathbf{T^i} = [t^i_1, \dots, t^i_{L^i}]$ associated with $\mathcal{S}$.

Each source i produces a pair $(\mathbf{Z}^i, \mathbf{T}^i)$ with a consistent format $\mathcal{X}^i$ and no internal gaps, but the collection $\mathcal{M} = \{(\mathbf{Z}^i, \mathbf{T}^i)\}_{i=1}^{S}$ may exhibit every irregularity introduced above. Splitting the measurements $\mathcal{S} = \{\mathbf{Z}^i\}$ from their timestamps $\mathcal{T}_S = \{\mathbf{T}^i\}$ simplifies the forecasting setup and unifies previous treatments.

A single $(\mathbf{Z}^i, \mathbf{T}^i)$ is itself an MTS, one whose only irregularity is uneven sampling of complete states $z^i_t \in \mathcal{X}^i$. In Section 3 we encode each pair with an independent, time-aware RNN; we call the corresponding data source *sensor*, although it does not need to correspond to a physical sensor.

The domain $\mathcal{X}^i$ reflects the data type: for tabular streams $\mathcal{X}^i \subset \mathbb{R}^{K^i}$ (e.g. wind speed & direction, $K^i = 2$); for images $\mathcal{X}^i \subset \mathbb{R}^{H^i \times W^i \times K^i}$ (e.g. satellite cloud maps). This work focuses on the tabular case.

We term *irregularities* any property that prevents stacking the sequences into a single tensor of shape $L \times \text{shape}(\mathcal{X})$, such as misalignment, missing data, or heterogeneous formats. Rather than treating an MTS as a uniformly sampled trace of a dynamical system $\mathcal{D}_t$, we view it as a set of measurement sequences $\mathcal{S}$ with their own timestamp sets $\mathcal{T}_S$, sampled, possibly irregularly, from $\mathcal{D}_t$.

2.2. Forecasting multivariate time series

For a given MTS $\mathcal{M} = (\mathcal{S}, \mathcal{T}_S)$, we consider an observer with access to limited information in the form of a contextual MTS. The observer's goal is to estimate the values of the system $\mathcal{D}_t$ on a set of target timestamps.

Definition 2 (Regression task on MTS). Given a context MTS $\mathcal{M}_c$ and a target MTS $\mathcal{M}_f = (\mathcal{S}_f, \mathcal{T}_{S_f})$, the regression task on MTS consists of finding a function f such that $f(\mathcal{M}_c, \mathcal{T}_{S_f}) = \mathcal{S}_f$.

This definition covers MTS regression for any type of irregularity. In this work, we are particularly interested in the forecasting task. Given an observer at time t_ϕ, the goal is to forecast the behavior of certain variables based on the data available at t_ϕ.

A subtle aspect of this problem is that the observer, at t_ϕ, does not know when or even if the events in $\mathcal{S}_f$ will occur. However, the task is not to predict the occurrence of these events, but rather to predict the values they would take if they were to occur at predefined timestamps $\mathcal{T}_{S_f}$. In other words, the objective is to predict the state of the underlying continuous system $\mathcal{D}_t$ at specific points in time using only the limited information available in $\mathcal{M}_c$. This represents a key distinction between Temporal Point Processes (TPPs) [16] and MTS forecasting, where the former focuses on predicting the event occurrence itself.

A general procedure for extracting $\mathcal{M}_c$ and $\mathcal{M}_f$ is as follows. Let t_ϕ denote the observer's time, and define three parameters: C^i, F^i, and $O^i \in \mathbb{R}$, representing the context length, forecast length, and offset time, respectively:

$$\mathcal{M}_c = (\mathcal{S}_c, \mathcal{T}_{S_c}), \tag{1}$$

$$\mathcal{S}_c = \{\, z^i_t \in \mathcal{S} \mid t_\phi - C^i \le t < t_\phi \,\}, \tag{2}$$

$$\mathcal{T}_{S_c} \vdash \mathcal{S}_c, \tag{3}$$

$$\mathcal{M}_f = (\mathcal{S}_f, \mathcal{T}_{S_f}), \tag{4}$$

$$\mathcal{S}_f = \{\, z^i_t \in \mathcal{S} \mid t_\phi \le t < t_\phi + F^i \,\}, \tag{5}$$

$$\mathcal{T}_{S_f} \vdash \mathcal{S}_f, \tag{6}$$

where $\vdash$ represents the mapping between events and their corresponding timestamps.

Fig. 2 illustrates this process. Reducing C^i is often motivated by the computational cost of processing long context windows, and by the fact that in many systems, the autocorrelation between states diminishes over time. Note that no event must occur precisely at t_ϕ.

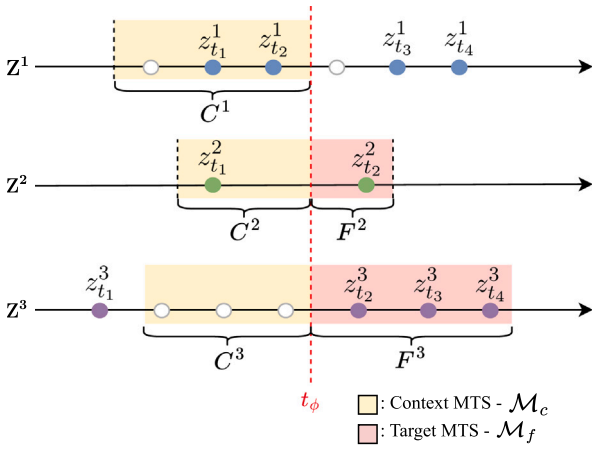


Fig. 2. An efficient method to extract context and forecast window pairs from an MTS involves defining C^i and F^i. In this example, sensor 3 has an empty context window due to missing data, and sensor 1 is only used for context, i.e., $F^1 = 0$.

This approach differs from traditional formulations, where MTS is treated as a grid-like structure. In such cases, $t_\phi \in \mathbb{Z}$ represents the step from which forecasting begins, and the last C steps are used to predict the next F steps. As detailed above, this approach is inadequate for irregular MTS, while our method embraces the complexities of asynchronous data, offering a more flexible solution to the forecasting task.

2.3. RNNs and GNNs as restrictions on $\mathcal{F}$

In supervised learning we approximate f by $f_\theta \in \mathcal{F}$, the architecture-defined function class parameterised by $\theta \in \Theta$; suitable parameterisation ensures universal approximation [17]. Many neural architectures can be viewed as *symmetry-restricted* optimisers: CNNs encode translation, gated RNNs and Transformers encode temporal or attentional symmetries, etc. [18].

For time series, strong temporal correlations and noise mean an unconstrained $\mathcal{F}$ (e.g. an MLP) overfits quickly [1]. We attribute the poor long-horizon performance of Transformers relative to simple linear models [19] to this lack of inductive bias—compounded by quadratic memory in sequence length.

RNNs [20] maintain a compact hidden state with *constant* memory, which is suitable for long sequences. Fig. 3 (left) shows a generic block; the encoder–decoder setup on the right processes a window $[z_{t_\phi-3}, z_{t_\phi-2}, z_{t_\phi-1}]$ into $h_{t_\phi-1}$, then predicts $[\hat{z}_{t_\phi}, \hat{z}_{t_\phi+1}]$.

We adopt the GRU [21]. Gated RNNs are quasi-invariant to *time warpings*, i.e. monotone differentiable maps $\tau : \mathbb{R}^+ \to \mathbb{R}^+$: for any f_θ and τ there exists $f_{\theta'} \in \mathcal{F}$ that processes $\mathbf{Z} \circ \tau$ identically [22]. This holds while τ is continuous; discontinuities induced by missing windows break the assumption, motivating richer RNN variants. Recent state-space models such as S4 and Mamba approach Transformer accuracy with linear memory [23,24], reinforcing RNNs as strong baselines for time-series tasks.

To exchange information across sensors $\mathbf{Z}^i_t$ we employ a Graph Neural Network [25]. Message passing at layer k is

$$\mathbf{x}^k_i = \gamma^k\left(\mathbf{x}^{k-1}_i, \square_{j \in \mathcal{N}(i)} \phi^k\left(\mathbf{x}^{k-1}_i, \mathbf{x}^{k-1}_j\right)\right), \tag{7}$$

where ϕ^k is the message function, $\square$ a permutation-invariant aggregator, and γ^k the update. This permutation symmetry ensures isomorphic graphs share representations, again constraining $\mathcal{F}$. We choose the Heterogeneous Graph Attention Network (HGAT) [26], treating sensors and relations as distinct node and edge types (see Section 3). A survey of GNN variants appears in [27].

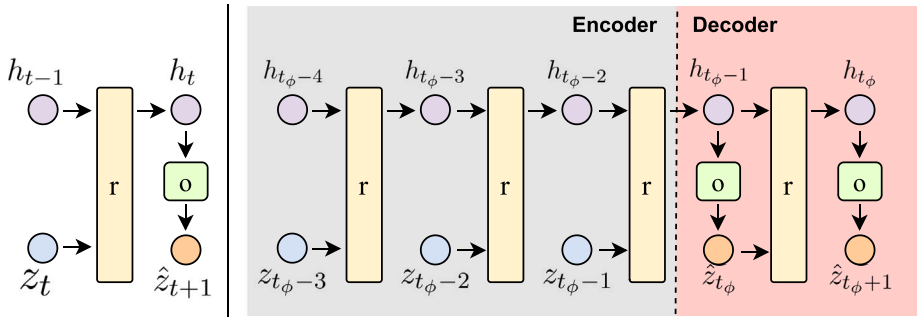


Fig. 3. Basic RNN block (left). RNN in an auto-regressive setup for a sequence-to-sequence task on an evenly spaced time series (right).

2.4. FPSO dynamics and irregularities in MTS

Although the contributions of this work apply to a general MTS context, it was developed to address the challenge of maintaining stable positions of floating platforms in deep water. FPSO units have been used for years to safely extract offshore oil reserves. These floating platforms enable the extraction and storage of crude oil while being held in place using mooring lines anchored to the ocean floor. Mooring systems play a critical role in ensuring positional stability, personnel safety, and smooth operation of various operations on a platform, such as extraction, production, and oil offloading.

The constant exposure of floating structures to offshore environmental conditions such as waves, currents, and wind leads to continuous stresses and strains on the platforms. Consequently, the structural integrity and mooring lines of these platforms degrade over time. Understanding and modeling the complex dynamics of the floating platform allow real-time detection of unexpected oscillatory patterns, which can help to assess the integrity of the mooring lines. In this work, we analyze all six degrees of freedom (DoF) of FPSO motion: surge, sway, and heave, which are translations; heading, roll, and pitch, which are rotations. Fig. 4 depicts all six time series. Note the differences in regularity and periodicity between high-frequency DoFs (pitch, roll, and heave) and low-frequency ones (heading, surge, and sway). The high-frequency DoFs exhibit much lower damping effects, and consequently more pronounced oscillatory patterns.

The ever increasing amount of data generated by sensors can be used to train ML algorithms for this task, but the same environmental characteristics that deteriorate the mooring system also affect the sensors, which are reflected in the measurements. Typical sensor data from FPSOs suffer from long missing data windows, different polling rates between sensors, and other deformities.

3. The gap-ahead multivariate time series regressor

Building on the concepts from Section 2, we introduce an architecture for the MTS forecasting task for $\mathcal{M} = (\{\mathbf{Z^i}\}_{i=0}^{|S|}, \{\mathbf{T^i}\}_{i=0}^{|S|})$. We call this the ***Gap-Ahead Multivariate Time Series Regressor*** (GAMR). GAMR's design is motivated by the need to handle individual sensor dynamics, irregular sampling, and inter-sensor interactions within an MTS.

GAMR is built upon three core components:

1. **Independent Sensor Encoding:** Each sensor's data stream $(\mathbf{Z^i}, \mathbf{T^i})$ is initially processed by an independent sequence model, capturing its unique temporal characteristics.

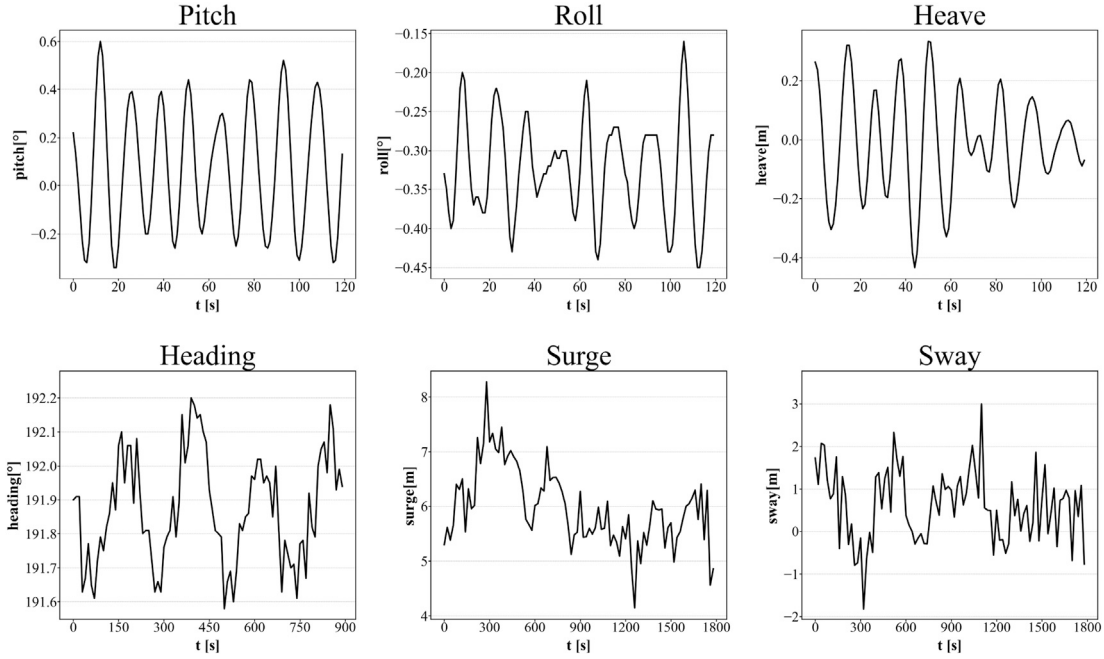


Fig. 4. Example windows of all six time series. High-frequency motions (pitch, roll and heave) display lower damping effects and more pronounced and well-behaved oscillatory patterns, while low-frequency (heading, surge, sway) exhibit high level of noise and damping effects.

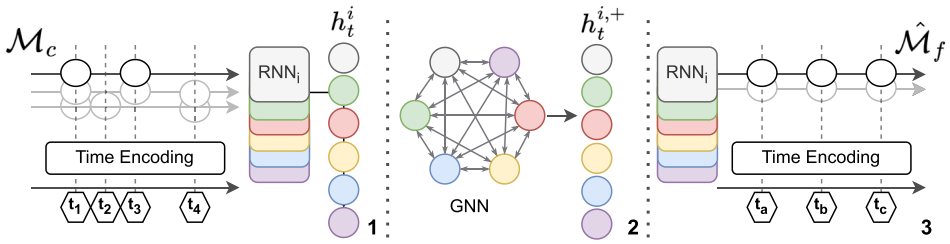


Fig. 5. The GAMR architecture overview. 1) Measurements are enriched with encoded representations of future timestamps. Each time series is encoded by an independent sequence model (here, an RNN). 2) A Regularized Heterogeneous GAT updates the node representation $h_{i,t}$ based on its neighborhood. 3) The updated representation $h_t^{i,+}$ initializes an autoregressive decoding process.

2. ***Gap-Ahead* Time Encoding (GATE):** A novel temporal encoding mechanism that explicitly informs the model about irregular time intervals between measurements, enabling flexible forecasting across potentially varying sampling rates.
3. **Information Diffusion via Heterogeneous Graph Attention Network (HGAT):** Based on [26,28], this component facilitates information exchange between sensors, modeled as different node types in a graph, enriching each sensor's representation with contextual information from others. This heterogeneous graph approach has proven effective for propagating information among encoded time series [29].

Fig. 5 illustrates the GAMR architecture's forward pass, which proceeds in three stages.

- **Stage 1: Independent Encoding with GATE.** Each sensor's context window $(\mathbf{Z_c^i}, \mathbf{T_c^i})$ is processed by its dedicated sequence model. While various sequential models could be used, our experiments utilize RNNs. Crucially, this stage incorporates the GATE mechanism, which encodes the time gap to the *next* event. The output is a fixed-size, time-informed hidden representation $h_t^i \in \mathbb{R}^H$ for each sensor i.
- **Stage 2: Information Diffusion via HGAT.** The hidden representations h_t^i from Stage 1 serve as initial node features in a Heterogeneous Graph Attention Network (HGAT) with g layers. We construct a fully-connected graph where each node represents a sensor i, and crucially, each sensor is assigned a unique node type. This setup defines a distinct relation type $r \in \mathcal{R}$ for every ordered pair of sensors (i, j), resulting in $|S|^2$ relation types. The HGAT mechanism [26] is specifically designed to handle such heterogeneity by learning a separate attention mechanism or message-passing function for each relation type r. Applying g layers of graph convolutions allows the model to approximate complex interactions tailored to each specific sensor pair (i, j). We use a fully-connected graph topology intentionally, treating the HGAT primarily as a powerful mechanism for global information propagation among all sensors, rather than leveraging explicit, pre-defined structural relationships which may not exist or be known. This process yields enriched representations $h_t^{i,+}$ that integrate context from all other sensors. While other information diffusion techniques could potentially be employed at this stage, exploring them is beyond the scope of this paper.
- **Stage 3: Autoregressive Decoding.** Each sensor's sequence model (e.g., RNN) is initialized with the enriched state $h_t^{i,+}$. It then generates hidden states h_t^i sequentially for the target timestamps $\mathbf{T_f^i}$ using the GATE mechanism in an autoregressive manner. For each target time $t \in \mathbf{T_f^i}$, a sensor-specific linear layer (W^i, b^i) maps the hidden state to the prediction $\hat{z}_t^i$:

$$\hat{z}_t^i = W^i h_t^i + b^i. \tag{8}$$

During decoding, the prediction $\hat{z}_t^i$ (as $z_{t\_}^i$ for the next step) and the state h_t^i (as $h_{t\_}^i$) are fed back into the sequence model to predict the value at the subsequent target timestamp, continuing until all values for $\mathbf{T_f^i}$ are estimated.

3.1. The gap-ahead time encoding mechanism

The GATE mechanism, detailed in Fig. 6, operates as follows. First, it encodes the relative time $t' = t - t_\phi$, where t_ϕ is a reference timestamp, using a time encoding function $\mathcal{G} : \mathbb{R} \to \mathbb{R}^{H_t}$. This yields negative t' for past events and positive t' for future events relative to t_ϕ, providing temporal context and a common clock across sensors. Note that $\forall t' \in \mathbf{T_c^i}$, $t' < 0$ and $\forall t' \in \mathbf{T_f^i}$, $t' > 0$ if t_ϕ marks the boundary between context $(\mathcal{M}_c)$ and forecast $(\mathcal{M}_f)$ windows.

Second, GATE concatenates the encoded time $\mathcal{G}(t')$ with the value of the *previous measurement* $z_{t\_}^i$ before feeding it into the sequence model:

$$h_t^i = \text{SeqModel}^i\left(z_{t\_}^i \;||\; \mathcal{G}(t'), h_{t\_}^i\right), \tag{9}$$

where $||$ denotes concatenation and SeqModel^i is the sensor-specific sequence model (e.g., RNN^i). This fundamentally reinterprets the sequence model's iteration. Unlike a standard RNN step (Fig. 3) which processes the *current* measurement z_t, the GATE-infused step uses the *previous* measurement $z_{t\_}^i$ and the *target* time t' encoding $\mathcal{G}(t')$. The model learns a transformation that integrates the information from $z_{t\_}^i$ into the hidden state $h_{t\_}^i$ *and* and evolves this state forward in time from $t\_$ to t. This dynamic adjustment of the time step resembles numerical ODE integration methods.

For the time encoding function $\mathcal{G}$, our experiments use the standard sinusoidal positional encoding (PE) from Ref. [2] as the default:

$$\mathcal{G}(t')_{2\,k} = \sin\left(\frac{t'}{1000^{2\,k/\tau}}\right), \quad \text{for even indices,} \tag{10}$$

$$\mathcal{G}(t')_{2\,k+1} = \cos\left(\frac{t'}{1000^{2\,k/\tau}}\right), \quad \text{for odd indices.} \tag{11}$$

We opt for this nonparametric absolute time encoding (ATE) because of its straightforward nature and the minimal enhancements

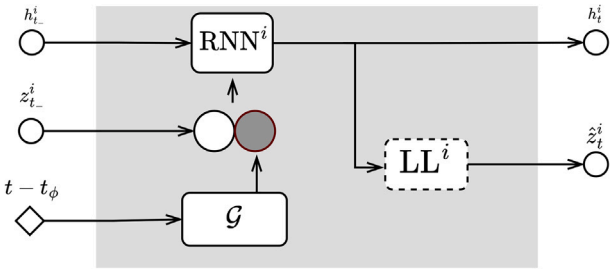


Fig. 6. GATE block setup. The transformation performed by the sequence model (here, an RNN) incorporates $z_{t\_}^i$ into $h_{t\_}^i$ and translates the hidden representation forward to time t. The resulting state h_t^i is projected by a linear layer (LL) to produce the prediction $\hat{z}_t^i$.

observed in previous research on fully learnable positional encoding alternatives over the nonparametric baseline [30]. However, we also tested Time2Vec [31], which is a parameterized ATE, in one of our model variants. It is worth mentioning that GATE can work with any ATE method. Although refining the selection of $\mathcal{G}$ could potentially boost performance, a detailed investigation is postponed for future work.

Importantly, although our experiments primarily utilize RNNs as the sequence processing backbone within GATE, the mechanism itself is general. It can potentially be applied to any sequential model that iteratively updates hidden states, such as Echo-State Networks [32] or Structured State Space models (S3) such as Mamba [23]. In this work, we focus on RNNs and defer the exploration of GATE with alternative sequence modeling architectures to future research.

The target timestamps $\mathcal{T}_f$ can be arbitrarily defined during inference. Section 5 demonstrates GATE's ability to generate forecasts at frequencies not encountered during training.

We train GAMR using supervised learning with the Adam optimizer [33]. Training batches are formed by uniformly sampling t_ϕ from the continuous time span of training data, not just from existing measurement timestamps ($z^i_{t_\phi}$ need not exist). This continuous sampling acts as dynamic data augmentation; multiple values t_ϕ can yield the same pair $(\mathcal{M}_c, \mathcal{M}_f)$ but with different relative time encodings $\mathcal{G}(t - t_\phi)$, promoting temporal robustness, similar to techniques *image translation* in computer vision [34]. Furthermore, we randomly remove context and target events for each sensor i with ratios $r^i_c, r^i_f \in [0, 0.6]$ (completely missing at random - MCAR), encouraging the model to learn to interpret $\mathbf{T^i}$ under varying data sparsity and improving temporal generalization.

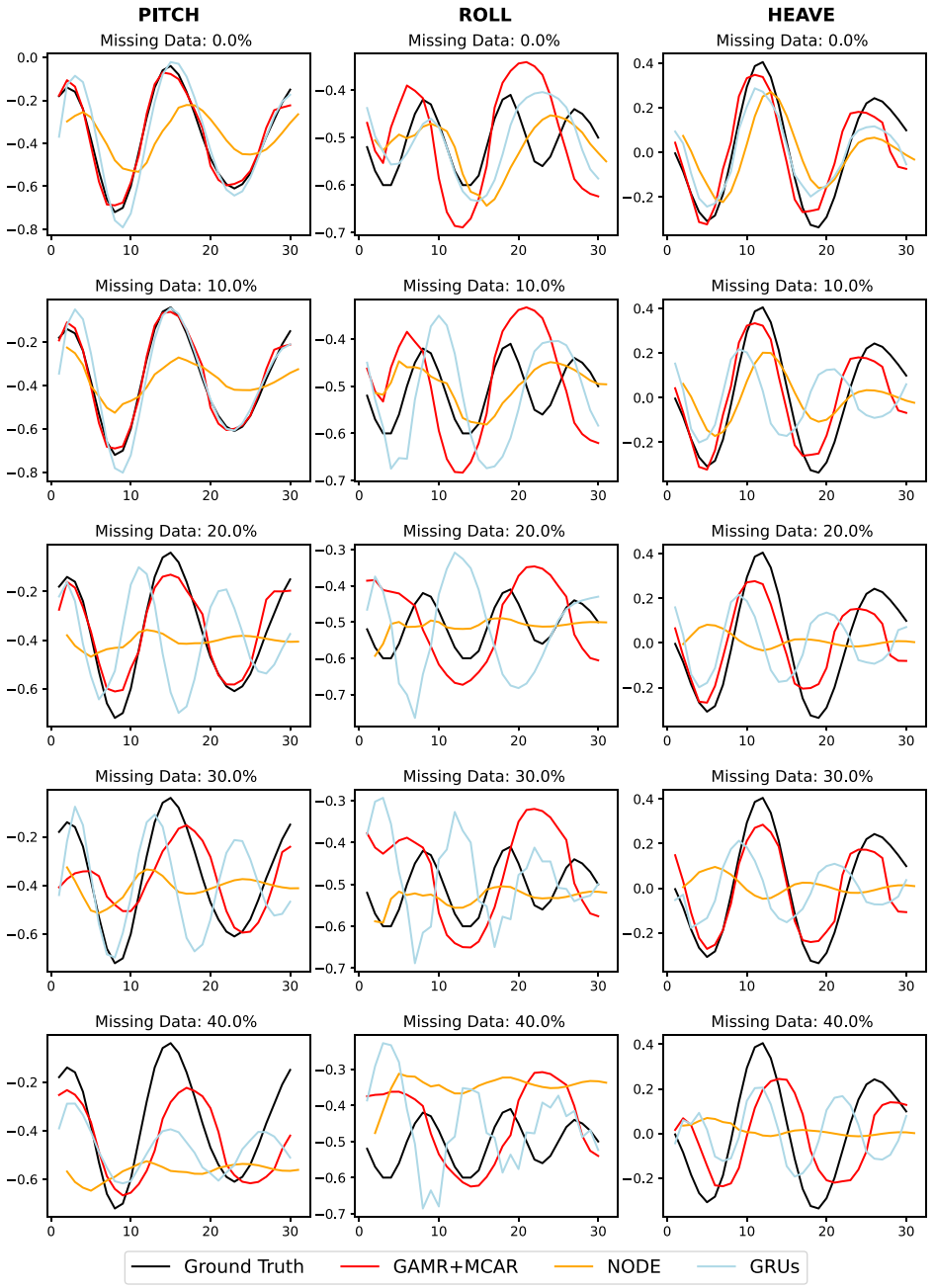


Fig. 7. Comparison between GAMR + MCAR, NODE, and GRUs variants of the model while forecasting the same window with distinct missing data proportions for the three high-frequency motions (Pitch, Roll, Heave). GAMR + MCAR maintains a more consistent prediction for all levels of missing data ratio.

GAMR consolidates and proposes several improvements over other prominent models. GRU-D, for example, also incorporates timestamps to address irregularities, but its masking strategy often leads to inefficient representations in highly irregular scenarios (see Section 2). Latent ODEs, in turn, provide a continuous-time framework for modeling MTS dynamics, yet they still require an encoder z to generate $z(t_0)$ that captures irregularities within the context window. In principle, the GAMR encoder could fulfill this role, producing $z(t_0)$ and enabling a Neural ODE to forecast the sequence. From this perspective, Neural ODEs are conceptually similar to the GATE mechanism. However, as demonstrated in our experiments (see Section 5) and noted in prior work, Neural ODEs are notoriously difficult to train, particularly when applied to noisy real-world signals such as FPSO attitude data collected directly from onboard sensors. GAMR mitigates these limitations by combining time-aware sequence modeling with graph-based aggregation, yielding a more robust and practical alternative for forecasting under challenging conditions.

Other recent end-to-end approaches, such as SeFT [13] and STraTS [35], also attempt to model irregular MTS without preprocessing. These methods share with GAMR the recognition that timestamp information must be integrated into latent representations. Nevertheless, they adopt an unordered set formulation, where each observation is represented as a timestamp–source–value triplet, and attention layers are applied to weigh triplets in the final representation. In contrast, GAMR treats MTS as sets of ordered sequences that can be independently encoded by any time-aware sequence model. This design exploits the natural inductive bias of temporal ordering, whereas SeFT and STraTS rely on computationally expensive attention mechanisms that compare all

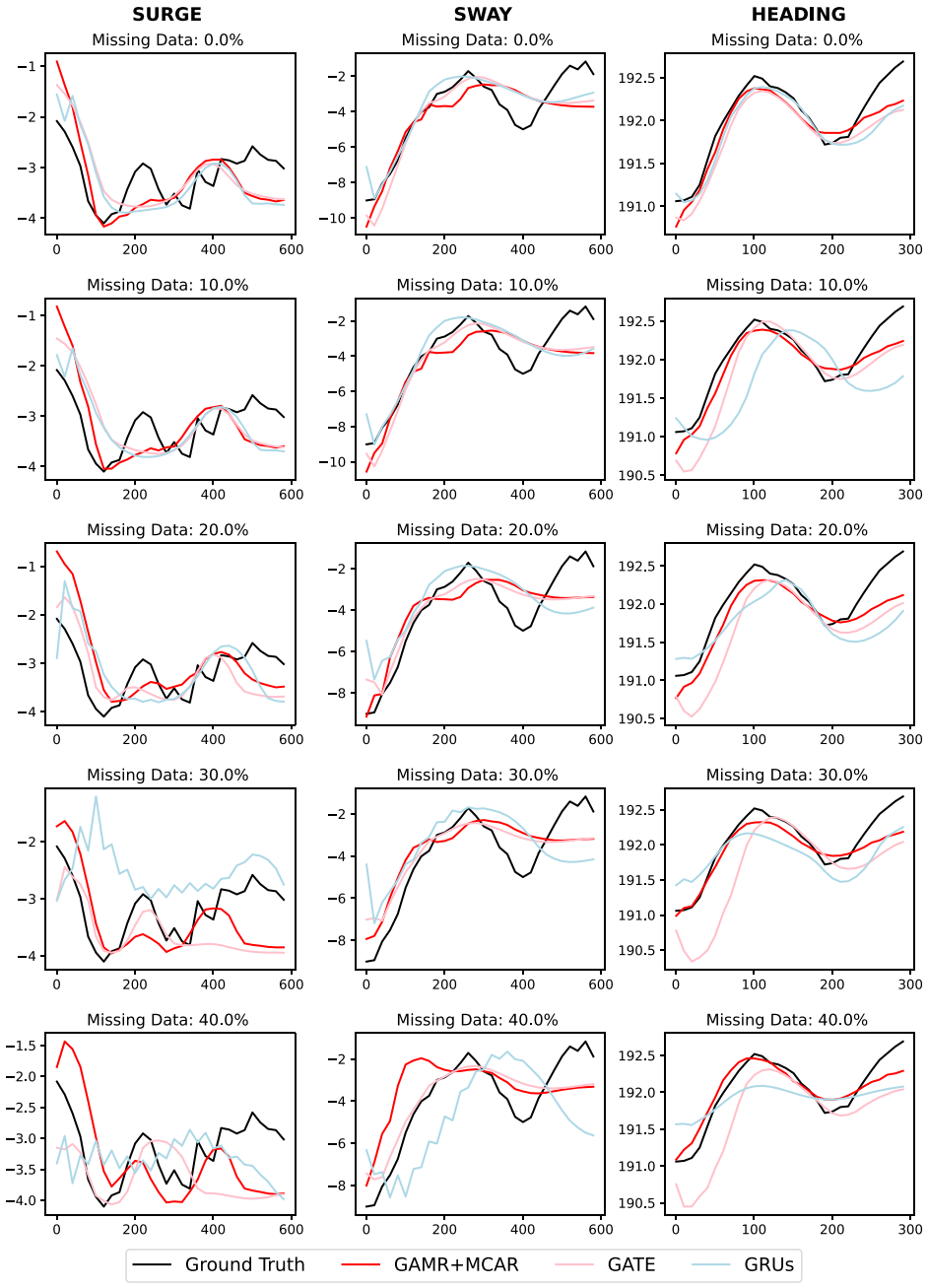


Fig. 8. Comparison between GAMR + MCAR, GATE, and GRUs variants of the model while forecasting the same window with distinct missing data proportions for the three low-frequency motions (Heading, Surge, Sway). Once again, GAMR + MCAR maintains a more consistent prediction for all levels of missing data ratio.

triplets pairwise. Furthermore, SeFT and STraTS were primarily developed for classification tasks and do not provide the flexible forecasting capabilities enabled by GAMR through the GATE mechanism.

4. Experimental setup

To evaluate the effectiveness of GAMR and its components, we performed experiments using real-world sensor data from an FPSO. The dataset comprises four years of measurements (2020-2023) from offshore sensors, with all variables originally sampled at 1 Hz. We designated the first three years (2020-2022) for training and the final year (2023) for testing. For a consistent evaluation across all models, we extracted 1387 context-forecast pairs $(\mathcal{M}_c, \mathcal{M}_f)$ from the 2023 test period and evaluated all models in the forecasting task defined by these pairs.

We compared five model variants to systematically assess the contribution of each architectural component.

- **GAMR + MCAR**: The complete proposed architecture, which incorporates both the GATE mechanism and the HGAT information diffusion stage, is trained with missing completely at random (MCAR) masking during training to enhance robustness to missing data.
- **GATE + MCAR**: An ablation variant that uses only the GATE mechanism (independent sensor encoding) without the HGAT component, also trained with MCAR masking. This assesses the impact of removing inter-sensor information diffusion.
- **GATE_T2V + MCAR**: Identical to **GATE + MCAR**, but substituting the default sinusoidal positional encoding with Time2Vec [31] as the time encoding function $\mathcal{G}$. This evaluates sensitivity to the choice of time encoder within GATE.
- **GATE**: The GATE mechanism without the HGAT component and without MCAR masking during training. This isolates the performance of GATE under standard training conditions.
- **GRUs + HGAT**: The HGAT component for information diffusion, but using a standard RNN encoder for each sensor *without* the GATE mechanism. This aims to evaluate the HGAT's contribution independently of the specialized time encoding.

Furthermore, we compared GAMR with two baseline models:

- **GRUs**: Standard Gated Recurrent Unit (GRU) networks process each sensor time series independently. This serves as a basic sequential modeling baseline without explicit handling of time gaps or inter-sensor communication.
- **Neural ODE**: Neural Ordinary Differential Equations [9], representing a continuous-time modeling approach. Standard Neural ODEs require an initial state h_0 to begin the integration process for forecasting. To provide an informed initial state that captures the context window's characteristics, including potential irregularities, we employed the following setup:
  - An encoder LSTM processed the context window $(\mathbf{Z_c^i}, \mathbf{T_c^i})$ for each sensor i.
  - At each step of this LSTM encoder, the input consisted of the concatenation of the measurement z_t^i and its corresponding timestamp t, where the timestamp was encoded using sinusoidal positional encoding (Equation (10)). This allows the encoder to implicitly learn about timing patterns and irregularities.
  - The final hidden state of the LSTM encoder for each sensor served as the initial state h_0 for the neural ODE solver.

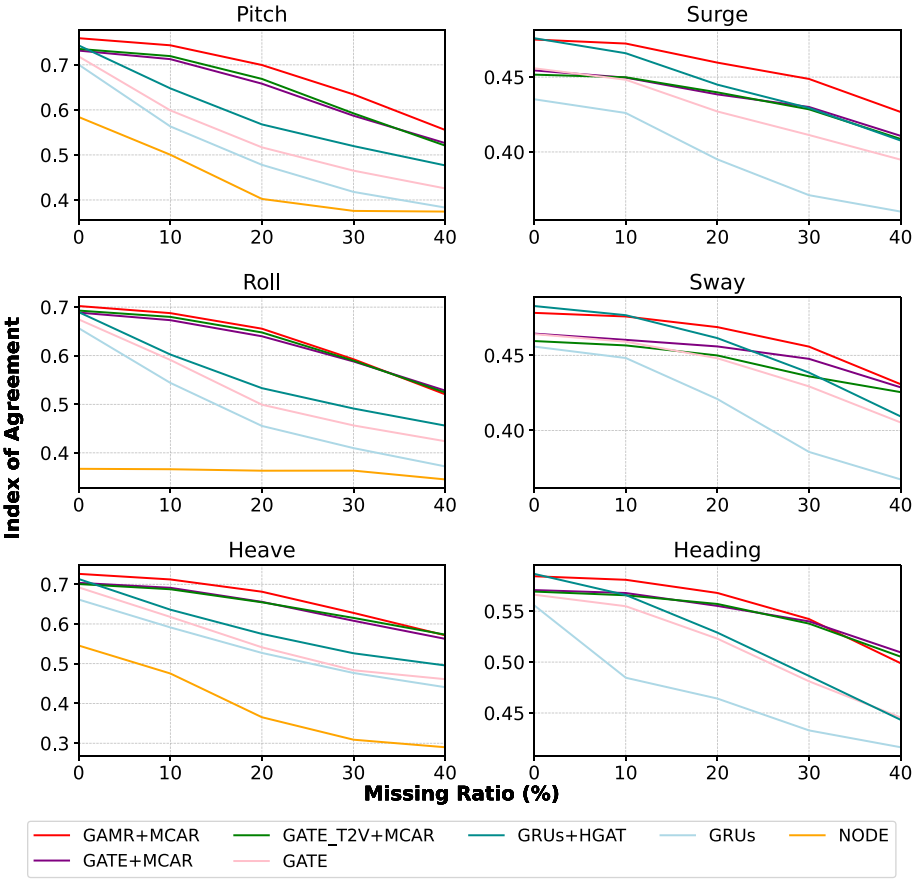


Fig. 9. Comparison of the IoA decay of all models as the missing data ratio increases from 0 % missing data to 40 %, across all of the six time series. Models with GATE that were trained with MCAR removals perform significantly better, exhibiting slower decay rates. Additionally, *GATE + MCAR* and *GATE_T2V + MCAR* have remarkably similar results.

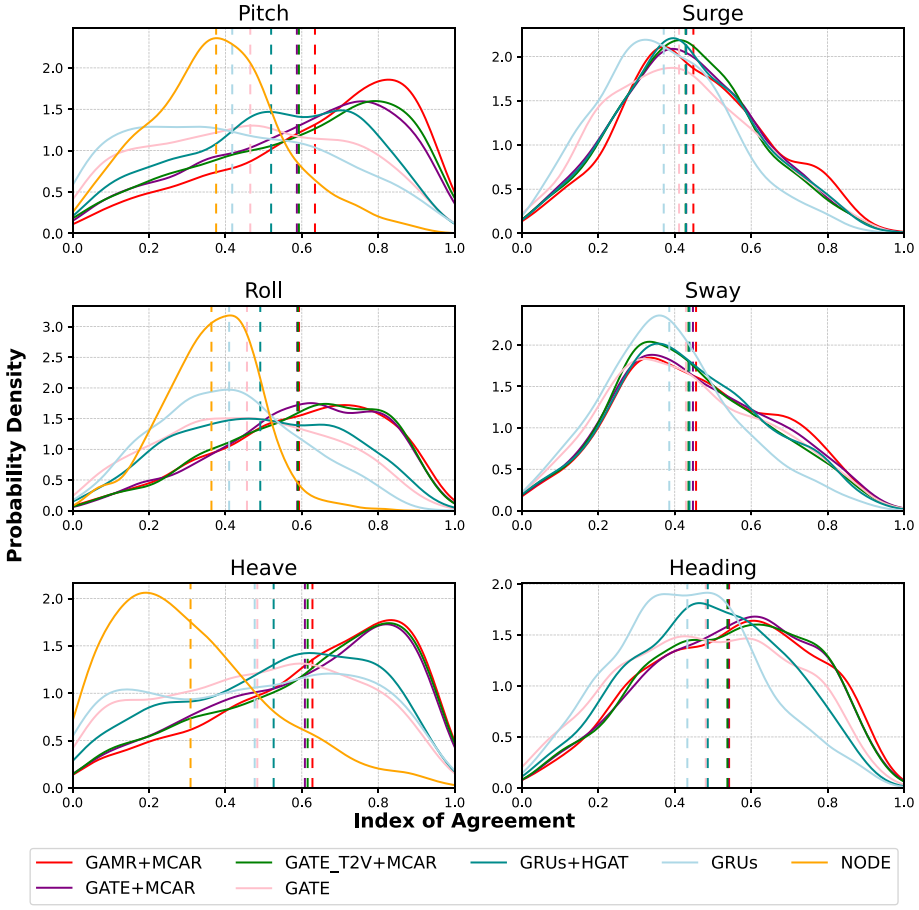


Fig. 10. Probability distribution of the IoA calculated for every forecast window in the test dataset on a 30 % missing data scenario. A distribution concentrated near 1.0 indicates a better model, since the likelihood of a forecast having a higher IoA is greater. Moreover, striped vertical lines indicate the median IoA for each model configuration. The GAMR + MCAR and GATE + MCAR variants produce forecasts with higher agreement values more often than others.

Table 1
Comparison of model performance (IoA, mean ± SD) across all fast DoFs (Pitch, Roll, Heave) from 0 to 40 % of missing data. Higher values indicate better performance.

Missing Ratio %	0	10	20	30	40
Pitch					
NODE	0.58 ± 0.24	0.50 ± 0.23	0.40 ± 0.20	0.38 ± 0.18	0.37 ± 0.16
GRUs	0.70 ± 0.23	0.56 ± 0.27	0.48 ± 0.27	0.42 ± 0.25	0.38 ± 0.23
GATE	0.72 ± 0.21	0.60 ± 0.26	0.52 ± 0.25	0.47 ± 0.25	0.43 ± 0.25
GATE + MCAR	0.73 ± 0.20	0.71 ± 0.21	0.66 ± 0.23	0.59 ± 0.24	0.53 ± 0.24
GRUs + HGAT	0.74 ± 0.20	0.65 ± 0.24	0.57 ± 0.25	0.52 ± 0.23	0.48 ± 0.22
GATE_T2V + MCAR	0.74 ± 0.20	0.72 ± 0.21	0.67 ± 0.23	0.59 ± 0.25	0.52 ± 0.25
GAMR + MCAR	**0.76 ± 0.19**	**0.74 ± 0.20**	**0.70 ± 0.22**	**0.63 ± 0.24**	**0.56 ± 0.25**
Roll					
NODE	0.37 ± 0.17	0.37 ± 0.17	0.36 ± 0.14	0.36 ± 0.13	0.35 ± 0.12
GRUs	0.66 ± 0.21	0.54 ± 0.23	0.46 ± 0.22	0.41 ± 0.19	0.37 ± 0.18
GATE	0.68 ± 0.20	0.59 ± 0.22	0.50 ± 0.23	0.46 ± 0.22	0.42 ± 0.21
GATE + MCAR	0.69 ± 0.19	0.67 ± 0.19	0.64 ± 0.20	**0.59 ± 0.21**	**0.53 ± 0.21**
GRUs + HGAT	0.69 ± 0.20	0.60 ± 0.22	0.53 ± 0.23	0.49 ± 0.22	0.46 ± 0.20
GATE_T2V + MCAR	0.69 ± 0.19	0.68 ± 0.19	0.65 ± 0.20	**0.59 ± 0.21**	**0.52 ± 0.21**
GAMR + MCAR	**0.70 ± 0.18**	**0.69 ± 0.19**	**0.66 ± 0.20**	**0.59 ± 0.22**	**0.52 ± 0.22**
Heave					
NODE	0.55 ± 0.27	0.48 ± 0.26	0.37 ± 0.24	0.31 ± 0.20	0.29 ± 0.19
GRUs	0.66 ± 0.25	0.59 ± 0.26	0.53 ± 0.28	0.48 ± 0.27	0.44 ± 0.26
GATE	0.69 ± 0.22	0.62 ± 0.25	0.54 ± 0.26	0.48 ± 0.26	0.46 ± 0.25
GATE + MCAR	0.70 ± 0.23	0.69 ± 0.23	0.66 ± 0.24	0.61 ± 0.25	0.56 ± 0.26
GRUs + HGAT	0.71 ± 0.22	0.64 ± 0.24	0.58 ± 0.25	0.53 ± 0.25	0.50 ± 0.24
GATE_T2V + MCAR	0.70 ± 0.22	0.69 ± 0.23	0.66 ± 0.24	0.62 ± 0.25	**0.57 ± 0.25**
GAMR + MCAR	**0.73 ± 0.22**	**0.71 ± 0.22**	**0.68 ± 0.23**	**0.63 ± 0.24**	**0.57 ± 0.25**

Given the challenges of applying standard Neural ODEs directly to the full MTS with significant irregularities without relying on masking or imputation techniques (as discussed in Section 2), we evaluated this baseline only on the three faster, more regularly sampled movements (heave, roll, and pitch) where inter-measurement intervals are consistent.

It is important to note that neither baseline inherently handles the full complexity of irregular MTS forecasting across all sensors without specific adaptations (such as imputation or masking). Therefore, they serve as valuable reference points but are not direct substitutes for the GAMR architecture, which is explicitly designed to manage these irregularities and facilitate intersensor communication.

We assess performance using the Index of Agreement (IoA) [36], a standardized metric comparing sequences $\hat{\mathbf{Z}}^{\mathbf{i}}$ and $\mathbf{Z}^{\mathbf{i}}$:

$$\mathrm{IoA}(\hat{\mathbf{Z}}^{\mathbf{i}}, \mathbf{Z}^{\mathbf{i}}) = 1 - \frac{\sum_t (\mathbf{Z}^{\mathbf{i}} - \hat{\mathbf{Z}}^{\mathbf{i}})^2}{\sum_t (|\hat{\mathbf{Z}}^{\mathbf{i}} - \bar{\mathbf{Z}}^{\mathbf{i}}| + |\mathbf{Z}^{\mathbf{i}} - \bar{\mathbf{Z}}^{\mathbf{i}}|)^2} \tag{12}$$

Ranging from 0 (no agreement) to 1 (perfect match), IoA robustly measures accuracy, accounting for systematic biases and data variability, and is common in hydrology. The training loss minimizes 1 − IoA averaged over target sensors $|S_f|$ with non-empty forecast windows:

$$\mathcal{L}(\hat{\mathcal{M}}_f, \mathcal{M}_f) = \frac{1}{|S_f|} \sum_i (1 - \mathrm{IoA}(\hat{\mathbf{Z}}^{\mathbf{i}}, \mathbf{Z}^{\mathbf{i}})) \tag{13}$$

Our first experiment tested model robustness to missing data by randomly masking 0 %, 10 %, 20 %, 30 %, and 40 % of test set points. This simulates sensor failures or communication gaps, assessing accuracy degradation as data availability decreases.

Our second experiment evaluated temporal generalization by producing forecasts at frequencies unseen during training. We altered target timestamps $\mathbf{T}_{\mathbf{f}}^{\mathbf{i}}$ to generate sequences at 0.2 Hz, 0.5 Hz, 2 Hz, and 5 Hz (frequencies > 1 Hz were absent during training). This assesses if the time-encoding mechanism $\mathcal{G}$ enables the model f_θ to approximate the underlying continuous system D_t, rather than just the sampled MTS $\mathcal{M}$.

Table 2
Comparison of model performance (IoA, mean ± SD) across all slow DoFs (Surge, Sway, Heading) from 0 to 40 % of missing data. Higher values indicate better performance.

Missing Ratio %	0	10	20	30	40
Surge					
GRUs	0.44 ± 0.19	0.43 ± 0.19	0.40 ± 0.18	0.37 ± 0.17	0.36 ± 0.16
GATE	0.46 ± 0.20	0.45 ± 0.20	0.43 ± 0.20	0.41 ± 0.20	0.40 ± 0.19
GATE + MCAR	0.45 ± 0.19	0.45 ± 0.19	0.44 ± 0.19	0.43 ± 0.19	0.41 ± 0.18
GRUs + HGAT	**0.48 ± 0.20**	**0.47 ± 0.20**	0.45 ± 0.19	0.43 ± 0.18	0.41 ± 0.17
GATE_T2V + MCAR	0.45 ± 0.19	0.45 ± 0.19	0.44 ± 0.19	0.43 ± 0.18	0.41 ± 0.18
GAMR + MCAR	**0.48 ± 0.20**	**0.47 ± 0.20**	**0.46 ± 0.20**	**0.45 ± 0.19**	**0.43 ± 0.19**
Sway					
GRUs	0.46 ± 0.21	0.45 ± 0.21	0.42 ± 0.20	0.39 ± 0.18	0.37 ± 0.16
GATE	0.46 ± 0.21	0.46 ± 0.21	0.45 ± 0.21	0.43 ± 0.21	0.41 ± 0.20
GATE + MCAR	0.46 ± 0.21	0.46 ± 0.21	0.46 ± 0.21	0.45 ± 0.21	**0.43 ± 0.20**
GRUs + HGAT	**0.48 ± 0.21**	**0.48 ± 0.21**	0.46 ± 0.21	0.44 ± 0.20	0.41 ± 0.18
GATE_T2V + MCAR	0.46 ± 0.21	0.46 ± 0.21	0.45 ± 0.21	0.44 ± 0.20	0.43 ± 0.19
GAMR + MCAR	**0.48 ± 0.21**	**0.48 ± 0.21**	**0.47 ± 0.21**	**0.46 ± 0.21**	**0.43 ± 0.20**
Heading					
GRUs	0.56 ± 0.22	0.49 ± 0.23	0.46 ± 0.23	0.43 ± 0.19	0.42 ± 0.17
GATE	0.57 ± 0.21	0.56 ± 0.21	0.52 ± 0.22	0.48 ± 0.22	0.45 ± 0.22
GATE + MCAR	0.57 ± 0.21	0.57 ± 0.21	0.56 ± 0.21	**0.54 ± 0.21**	0.51 ± 0.20
GRUs + HGAT	**0.59 ± 0.22**	0.57 ± 0.22	0.53 ± 0.21	0.49 ± 0.20	0.44 ± 0.18
GATE_T2V + MCAR	0.57 ± 0.21	0.57 ± 0.21	0.56 ± 0.21	**0.54 ± 0.21**	**0.51 ± 0.21**
GAMR + MCAR	0.58 ± 0.22	**0.58 ± 0.22**	**0.57 ± 0.22**	**0.54 ± 0.21**	0.50 ± 0.22

Table 3
Wilcoxon test p-values for IoA distributions at 20 % missing data across the three fast DoFs. Lower values indicate superiority of the row model over the column model.

	Pitch		Roll		Heave	
	GRUs	NODE	GRUs	NODE	GRUs	NODE
GATE + MCAR	< 1e−230*	8.4e−139	< 1e−230*	9.9e−184	< 1e−230*	1.2e−101
GATE_T2V + MCAR	4.0e−103	< 1e−230*	1.7e−132	< 1e−230*	1.45e−63	< 1e−230*
GAMR + MCAR	3.3e−125	1.1e−164	4.0e−132	6.9e−189	4.7e−76	3.7e−120

* *underflow*.

Table 4
Wilcoxon test p-values for IoA distributions at 20 % missing data across the three slow DoFs. Lower values indicate superiority of the row model over the column model.

	Surge		Sway		Heading	
	GRUs	GATE	GRUs	GATE	GRUs	GATE
GATE + MCAR	< 1e−230*	3.45e−05	< 1e−230*	5.34e−04	< 1e−230*	1.97e−12
GATE_T2V + MCAR	1.21e−18	< 1e−230*	2.92e−09	< 1e−230*	8.08e−44	< 1e−230*
GAMR + MCAR	2.44e−30	1.36e−14	3.51e−22	4.68e−10	7.60e−51	3.92e−20

* *underflow*.

Using real FPSO sensor data, these experiments evaluate the architecture's robustness and flexibility against industrial dynamics and data irregularities. We hypothesize that time-informed model variants utilizing MCAR training will effectively handle missing data and variable sampling rates.

5. Results and analysis

Figs. 7 and 8 present an illustration of the forecast of each model for $\mathcal{S}_f$ values as missing data levels increase. It should be noted that *GAMR+MCAR* not only provides more accurate predictions but also maintains greater consistency between different missing-data ratios. In contrast, variant *GRUs* lacks a time encoding mechanism to detect data irregularities, leading to significantly different forecasts based on the extent of data removed from the context of MTS $\mathcal{M}_c$. Furthermore, the variant *GATE*, although equipped with a time encoding, struggled to deliver stable forecasts under missing data conditions, as it was not trained on artificially induced irregularities. In Fig. 8, it is important to note that *GATE* produces different predictions depending on the level of missing data. This occurs because of the lack of the MCAR procedure during training, which prevents the model from acquiring the ability to correctly incorporate timestamp information.

Fig. 9 shows the drop in IoA for the seven models as the proportion of missing data increases. Notably, time-sensitive models trained using MCAR (*GAMR+MCAR*, *GATE+MCAR*, and *GATE_T2V+MCAR*) show a much slower decline than the other models. Our tests also indicated that integrating HGAT improved model performance when fewer data were missing. This supports prior research findings that used similar

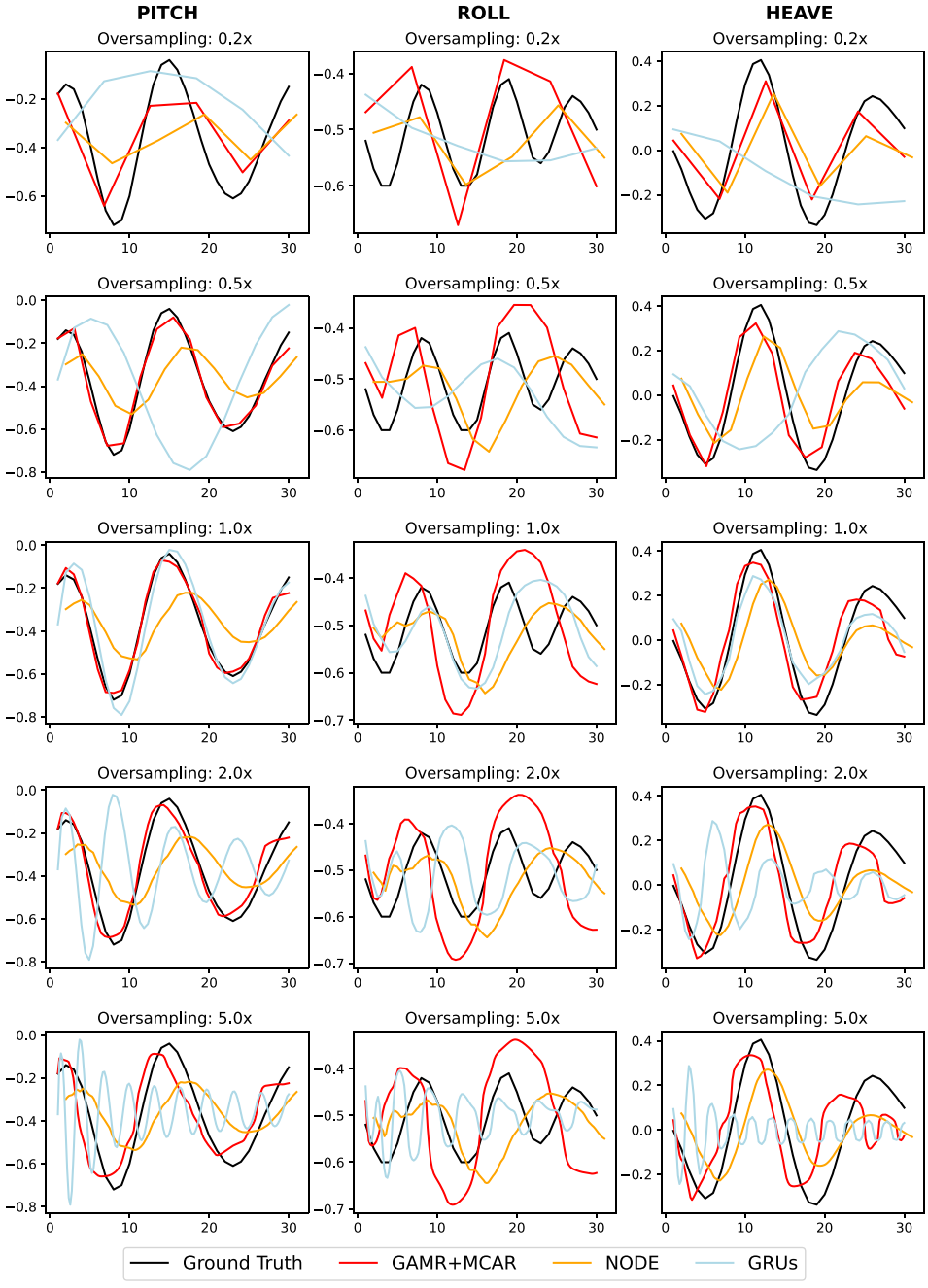


Fig. 11. Comparison between predictions of GRUs, GATE and GAMR+MCAR model variants when the target timestamps are built with different sampling rates for the high-frequency motions. Only GAMR+MCAR is able to adapt, including to frequencies unseen during training such as 2Hz and 5Hz.

HGAT methodologies [29]. These results suggest that the enhanced resilience to missing data is tied to linking GATE with MCAR and not to HGAT. The baseline *NODE* performs less well compared to the other models, particularly in predicting roll movement, noted for its complexity. We propose that this is due to the limitations in stability and expressivity inherent in the original formulation, as noted by several studies [37–39]. However, it is relevant to note that all the GAMR architecture variants tested exhibited a very well-behaved convergence regime.

Fig. 10 illustrates the IoA probability distribution in the case of 30 % data loss. In the left column, it is evident that the combination of GATE and MCAR methods more often yielded more accurate predictions, especially for high-frequency movements.

To quantitatively evaluate the results, statistical tests were performed. Tables 3 and 4 display Wilcoxon's test p-values comparing our main model variants to the baselines at an intermediate 20 % missing data level. Lower p-values suggest more significant model differences favoring the raw model. These results show unequivocally the superiority of the models equipped with GATE and MCAR in this setup.

The results of the initial experiment, which evaluated the models with escalating data missing rates, demonstrated that the variants *GAMR + MCAR*, *GATE + MCAR* and *GATE_T2V + MCAR* were significantly more accurate, particularly when faced with high levels of missing data. Although the HGAT module enhances the model, it is beneficial mainly at lower data loss rates. This aligns with previous findings, suggesting that while HGAT effectively propagates information between

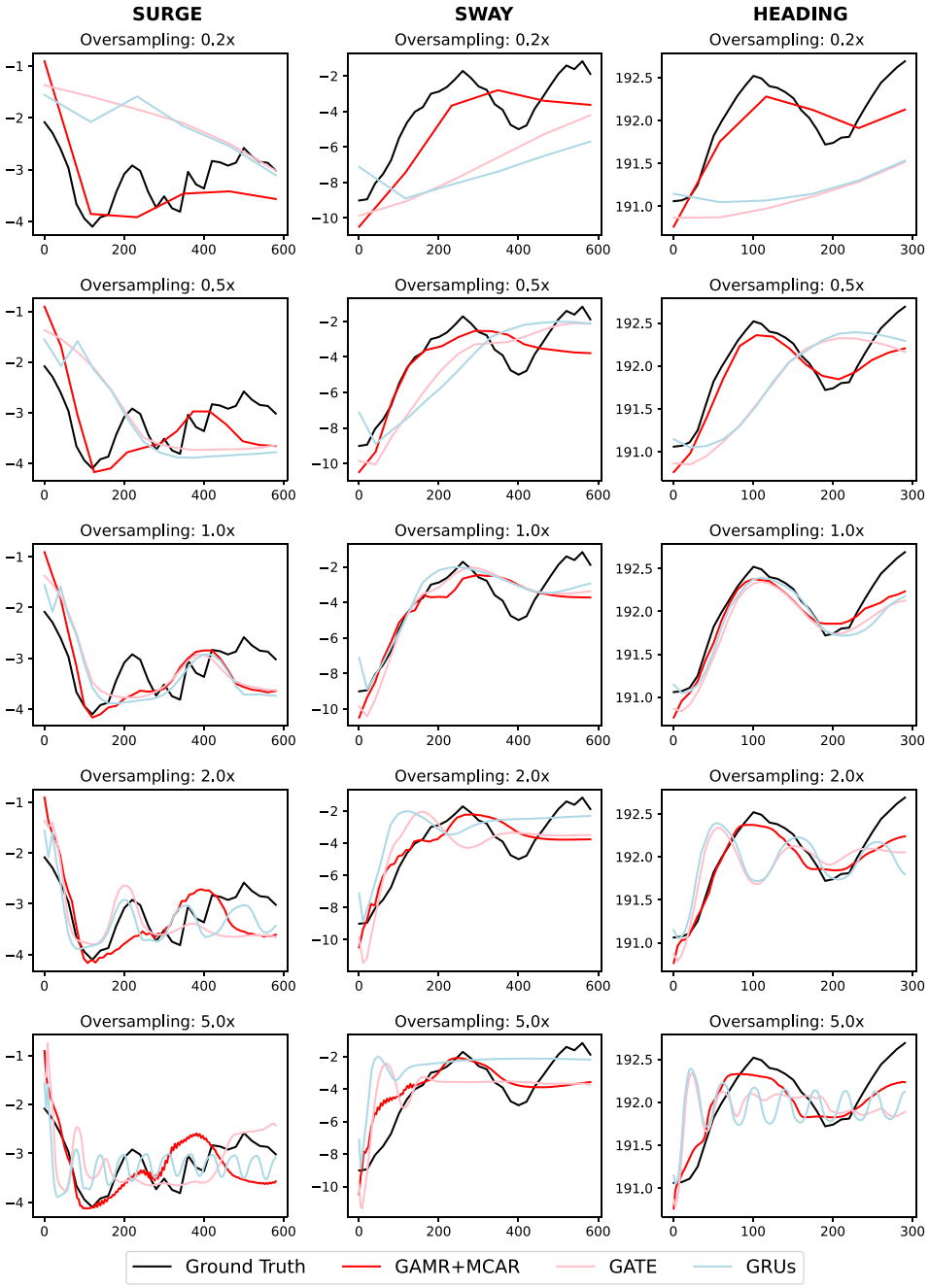


Fig. 12. Comparison between predictions of GRUs, Only GATE and GAMR + MCAR model variants when the target timestamps are built with different sampling rates for the low-frequency motions. Only GAMR + MCAR is able to adapt, including to frequencies unseen during training such as 2Hz and 5Hz.

sensors, it does not enhance the robustness of the model against missing data [29]. This is evident from the comparison of *GATE + MCAR* with *GAMR + MCAR* and *GRUs* with *GRUs + HGAT* (Tables 1 and 2).

Figs. 11 and 12 present the results for the second experiment, which changes the target timestamps $\mathcal{T}_f$ to verify if the model is able to generate predictions at unseen frequencies. As expected, models that were not time-informed and trained with MCAR were not able to generalize. On the other hand, our method produced sensible forecasts at unseen frequencies such as 2Hz and 5Hz.

This result provides another robust evidence that the combination of GATE and MCAR training induces the models to acquire this time generalization property by using the information coming from $\mathcal{G}(t-t_\phi)$ to dynamically evolve the state of the system over time, similar to NODEs.

6. Conclusion

Our results provide evidence that combining time-informed RNNs with a training method that introduces artificial irregularities enhances model robustness to missing data and induces temporal generalization. This work also shows that an HGAT can be integrated with RNNs and time encoders to address diverse MTS irregularities. Our main contribution is demonstrating that irregular MTS can be modeled effectively using independent, gap-ahead-informed encoders whose information is propagated within a common latent space via mechanisms like HGAT.

By adopting general definitions that account for data irregularities, we developed a simple, cost-effective, and robust architecture. This architecture learns a continuous representation of the underlying system D_t from sampled data and demonstrates the ability to adapt to sampling frequencies five times higher than the maximum seen during training. These results suggest the architecture's potential as a versatile model for various MTS applications.

The findings indicate that incorporating a time representation $\mathcal{G}(t-t_\phi)$ into the RNN input, combined with a simple MCAR mechanism during training, is sufficient to achieve this temporal generalization and the accompanying robustness to missing data.

We propose that the GATE mechanism could inspire alternative NODE formulations capable of implicitly representing system time derivatives. Future research directions include decomposing the GATE block into distinct transformations: a time translation function m where $m(h_{t\_}, t - t_i) = h_t$, and an ingestion function n where $n(h_t, z_t) = h_t^+$ (with h_t^+ being the refined hidden state after observing event z_t). This decomposition would enable imposing time-related symmetries on m and necessitates further study of the time-encoding function $\mathcal{G}$. Evaluating the length generalization capabilities of this approach, drawing parallels with findings on positional encoding in NLP tasks [40], presents another promising avenue.

CRediT authorship contribution statement

Marcel Barros: Writing – original draft, Software, Investigation, Formal analysis, Conceptualization. **Lucas P. Fontenele:** Writing – original draft, Software, Data curation. **Mariana S. Silva:** Writing – original draft, Software, Investigation. **Thiago Rissi:** Software, Investigation. **Eduardo Cabrera:** Software, Investigation. **Eduardo A. Tannuri:** Writing – review & editing, Validation, Supervision, Project administration. **Edson S. Gomi:** Writing – review & editing, Validation, Resources, Project administration. **Rodrigo A. Barreira:** Supervision, Resources, Project administration, Methodology. **Anna Helena Reali:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Marcel Rodrigues de Barros reports that financial support was provided by Petrobras. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code is available, however the dataset used is confidential, since it refers to measurements from actual moored floating platform and was provided by Petróleo Brasileiro S.A.

References

[1] W.W.S. Wei, Multivariate Time Series Analysis and Applications, Wiley Series in Probability and Statistics, Wiley, Hoboken, NJ, 2019.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017, https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[3] S. Makridakis, E. Spiliotis, V. Assimakopoulos, Statistical and machine learning forecasting methods: concerns and ways forward, PLoS One 13 (3) (2018) 1–26, Public Library of Science https://doi.org/10.1371/journal.pone.0194889

[4] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, O. Tabona, A survey on missing data in machine learning, J. Big Data 8 (1) (2021) 140, https://doi.org/10.1186/s40537-021-00516-9

[5] A.M. Saad, F. Schopp, A.N. Queiroz Filho, R.D.S. Cunha, I.H.F. Santos, R.A. Barreira, E.A. Tannuri, E.S. Gomi, A.H.R. Costa, FPSO mooring line failure detection based on predicted motion, in: Proceedings of the 40th International Conference on Ocean, Offshore and Arctic Engineering, American Society of Mechanical Engineers Digital Collection, 2021, https://doi.org/10.1115/OMAE2021-62413, https://asmedigitalcollection.asme.org/OMAE/proceedings-abstract/OMAE2021/85116/1121313

[6] J. Friend, A. Hauck, S. Kurada, T. Hartvigsen, C. Sen, E.A. Rundensteiner, Handling missing values in multivariate time series classification, in: 2018 IEEE MIT Undergraduate Research Technology Conference (URTC), 2018, pp. 1–3, https://doi.org/10.1109/URTC45901.2018.9244769

[7] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent neural networks for multivariate time series with missing values, Sci. Rep. 8 (1) (2018) 6085, https://doi.org/10.1038/s41598-018-24271-9

[8] D. Neil, M. Pfeiffer, S.-C. Liu, Phased LSTM: accelerating recurrent network training for long or event-based sequences, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 29, Curran Associates, Inc., 2016, https://proceedings.neurips.cc/paper_files/paper/2016/file/5bce843dd76db8c939d5323dd3e54ec9-Paper.pdf

[9] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, in: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018, https://arxiv.org/abs/1806.07366

[10] K.-I. Funahashi, Y. Nakamura, Approximation of dynamical systems by continuous time recurrent neural networks, Neural Networks 6 (6) (1993) 801–806, https://doi.org/10.1016/S0893-6080(05)80125-X, https://www.sciencedirect.com/science/article/pii/S089360800580125X

[11] A.L.I. Norcliffe, C. Bodnar, B. Day, J. Moss, P. Lio, Neural ODE processes: a short summary, in: The Symbiosis of Deep Learning and Differential Equations, 2021, https://openreview.net/forum?id=6yovcKE2LeN

[12] G. Karagiorgi, G. Kasieczka, S. Kravitz, B. Nachman, D. Shih, Machine learning in the search for new fundamental physics, Nat. Rev. Phys. 4 (6) (2022) 399–412, https://doi.org/10.1038/s42254-022-00455-1

[13] M. Horn, M. Moor, C. Bock, B. Rieck, K. Borgwardt, Set functions for time series, in: H. Daumé III, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, Vol. 119 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 4353–4363, https://proceedings.mlr.press/v119/horn20a.html

[14] S.N. Shukla, B. Marlin, Multi-time attention networks for irregularly sampled time series, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=4c0J6lwQ4

[15] Z. Huang, Y. Sun, W. Wang, Learning continuous system dynamics from irregularly-sampled partial observations, in: Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., 2020, pp. 16177–16187, https://arxiv.org/abs/2011.03880

[16] O. Shchur, A.C. Türkmen, T. Januschowski, S. Günnemann, Neural temporal point processes: a review, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4585–4593, https://doi.org/10.24963/ijcai.2021/623

[17] E.J. Hartman, J.D. Keeler, J.M. Kowalski, Layered neural networks with gaussian hidden units as universal approximations, Neural Comput. 2 (2) (1990) 210–215, https://doi.org/10.1162/neco.1990.2.2.210

[18] M.M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep Learning: grids, groups, graphs, geodesics, and gauges, arXiv:2104.13478 (May 2021) https://doi.org/10.48550/arXiv.2104.13478

[19] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, Proc. AAAI Conf. Artif. Intell. 37 (9) (2023) 11121–11128, https://doi.org/10.1609/aaai.v37i9.26317, https://ojs.aaai.org/index.php/AAAI/article/view/26317

[20] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536, https://doi.org/10.1038/323533a0

[21] K. Cho, B. van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder–decoder approaches, in: D. Wu, M. Carpuat, X. Carreras, E.M. Vecchi (Eds.), Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 103–111, https://doi.org/10.3115/v1/W14-4012, https://aclanthology.org/W14-4012/

[22] C. Tallec, Y. Ollivier, Can recurrent neural networks warp time?, in: International Conference on Learning Representations, 2018, https://openreview.net/forum?id=SJcKhk-Ab

[23] A. Gu, K. Goel, C. Re, Efficiently modeling long sequences with structured state spaces, in: International Conference on Learning Representations, 2022, https://openreview.net/forum?id=uYLFoz1vlAC

[24] A. Gu, T. Dao, Mamba: linear-time sequence modeling with selective state spaces, in: First Conference on Language Modeling, 2024, https://openreview.net/forum?id=tEYskw1VY2

[25] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80, https://doi.org/10.1109/TNN.2008.2005605

[26] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 2022–2032, event-place: San Francisco, CA, USA. https://doi.org/10.1145/3308558.3313562

[27] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2021) 4–24, https://doi.org/10.1109/TNNLS.2020.2978386

[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018, https://openreview.net/forum?id=rJXMpikCZ

[29] M. Barros, A. Pinto, A. Monroy, F. Moreno, J. Coelho, A.P. Silva, C.F.D. Netto, J.R. Leite, M. Mathias, E. Tannuri, A. Jordao, E. Gomi, F. Cozman, M. Dottori, A.H.R. Costa, Early detection of extreme storm tide events using multimodal data processing, Proc. AAAI Conf. Artif. Intell. 38 (20) (2024) 21923–21931, number: 20. https://doi.org/10.1609/aaai.v38i20.30194, https://ojs.aaai.org/index.php/AAAI/article/view/30194

[30] B. Wang, L. Shang, C. Lioma, X. Jiang, H. Yang, Q. Liu, J.G. Simonsen, On position embeddings in BERT, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=onxoVA9FxMw

[31] S.M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, M. Brubaker, Time2Vec: learning a vector representation of time, arXiv:1907.05321 (Jul. 2019) https://doi.org/10.48550/arXiv.1907.05321

[32] H. Jaeger, The echo state approach to analysing and training recurrent neural networks, GMD Report 148, GMD - German National Research Institute for Computer Science, 2001, https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf

[33] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015, http://arxiv.org/abs/1412.6980

[34] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, J. Big Data 6 (1) (2019) 60, https://doi.org/10.1186/s40537-019-0197-0

[35] S. Tipirneni, C.K. Reddy, Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series, ACM Trans. Knowl. Discov. Data 16 (6), New York, NY, USA, Association for Computing Machinery (Jul. 2022) https://doi.org/10.1145/3516367

[36] C.J. Willmott, ON the validation of MODELS, Phys. Geogr. 2 (2) (1981) 184–194, Taylor & Francis, https://doi.org/10.1080/02723646.1981.10642213

[37] E. Dupont, A. Doucet, Y.W. Teh, Augmented neural ODEs, in: H. Wallach, H. Larochelle, A. Beygelzimer, F.D. Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper_files/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf

[38] A. Norcliffe, C. Bodnar, B. Day, N. Simidjievski, P. Lió, On second order behaviour in augmented neural ODEs, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., 2020, pp. 5911–5921, https://proceedings.neurips.cc/paper_files/paper/2020/file/418db2ea5d227a9ea8db8e5357ca2084-Paper.pdf

[39] I.A. Auzina, C. Yildiz, S. Magliacane, M. Bethge, E. Gavves, Modulated neural ODEs, in: Thirty-Seventh Conference on Neural Information Processing Systems, 2023, https://openreview.net/forum?id=Op9z2QfXbC

[40] A. Kazemnejad, I. Padhi, K. Natesan, P. Das, S. Reddy, The impact of positional encoding on length generalization in transformers, in: Thirty-Seventh Conference on Neural Information Processing Systems, 2023, https://openreview.net/forum?id=Drrl2gcjzl