D^{2}ARIF: DELAY- AND DENSITY-AWARE REPRESENTATION LEARNING FOR IRREGULAR MULTIVARIATE TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

In irregular multivariate time series (IMTS), measurements are recorded asynchronously across variables, creating two challenges for forecasting. First, existing methods usually perform information interaction among temporally aligned or nearby observations, which makes cross-variable delayed interactions with long time intervals difficult to model directly. Second, existing methods usually treat irregular sampling as a difficulty in modeling, and lack the utilization of observation density information brought by irregular sampling. To address the above challenges, we propose **D**elay- and **D**ensity-**A**ware **R**epresentation Learning for **I**rregular Multivariate Time Series **F**orecasting (D^{2}ARIF) with two corresponding components: the Delay-Aware Module and Density-Aware Module. The Delay-Aware Module maintains a separate state for each source variable, decays these states according to the elapsed interval between active patches, and uses each target variable's query to weight information from previously observed sources. The Density-Aware Module estimates a continuous observation-time density for each variable and temporal patch using locally adaptive bandwidths, then encodes the resulting distributions for forecasting. Compared with state-of-the-art methods, D^{2}ARIF achieves improvements of up to 6.8% across four benchmarks.

1 INTRODUCTION

Irregular Multivariate Time Series (IMTS) occur in healthcare, biomechanics, climate science, and other domains where variables are measured independently. Different variables can be recorded at different times, and consecutive measurements can be separated by uneven intervals. Such variable asynchronicity and nonuniform sampling pose dual challenges to accurate prediction.

To address these challenges, existing methods have explored ways of representing irregular observations. mTAN Shukla & Marlin (2021) uses attention over reference times, whereas Neural ODEs Chen et al. (2018) and ContiFormer Chen et al. (2023) model temporal evolution in continuous time. Raindrop Zhang et al. (2022) models dependencies among sensors, while t-PatchGNN Zhang et al. (2024) and Hi-Patch Luo et al. (2025) focus on interactions within local temporal patches. Although these approaches provide different mechanisms for handling asynchronous observations and irregular intervals, two issues remain insufficiently addressed: capturing cross variable information over irregular time delays and representing observation time distributions within local regions.

The first issue concerns dependencies across variables over longer time intervals. An observation from one variable may remain useful for another after a substantial time gap, but existing attention and message passing methods tend to focus on dependencies between variables that are close in time, making it difficult to capture dependencies that persist over long time intervals. The second issue concerns local observation density. The same number of observations can be concentrated in a short interval or spread across a patch, resulting in different temporal distributions. The existing method HEADS Hu et al. (2026) measures changes in observation sparsity between the historical and forecasting windows, but its temporal observation density captures only window-level statistics and lacks the observation time distribution within each patch.

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

These issues motivate D^{2}ARIF, a delay- and density-aware representation learning framework for IMTS forecasting. The Delay-Aware Module preserves each source variable in a separate state, applies elapsed-time decay between active patches, and lets each target variable retrieve information from previously observed sources. The Density-Aware Module estimates observation time densities for each variable and patch from historical timestamps and encodes them before prediction. These modules form temporal and density representations that are fused for forecasting. The main contributions are summarized as follows:

- We develop a Delay-Aware Module that adapts delta-rule state updates to IMTS forecasting through source-specific states, elapsed-time decay, and weighting of source states separately for each target variable.
- We propose a plug-and-play Density-Aware Module that can be seamlessly inserted into existing forecasting models: it uses locally adaptive bandwidths to estimate observation-time densities for each variable and patch, then encodes them as variable-specific density representations for forecasting.
- We propose Delay- and Density-Aware Representation Learning for Irregular Multivariate Time Series Forecasting (D^{2}ARIF), which achieves state-of-the-art performance. Compared with state-of-the-art methods, D^{2}ARIF achieves improvements of up to 6.8% across four benchmarks.

2 Related Work

2.1 Handling Irregular Sampling in Multivariate Time Series

Alignment-based methods construct a shared temporal representation, patch-based methods aggregate observations within temporal regions, and timestamp-based methods retain individual observation times. IP-Net Shukla & Marlin (2019), mTAN Shukla & Marlin (2021), Warpformer Zhang et al. (2023), and KAFNet Zhou et al. (2026) align or compress observations on shared temporal supports. t-PatchGNN Zhang et al. (2024), Hi-Patch Luo et al. (2025), TimeCHEAT Liu et al. (2025), VIMTS Hu et al. (2025), and APN Liu et al. (2026b) construct local or adaptive patches. GRU-D Che et al. (2018), continuous-time models Rubanova et al. (2019); Kidger et al. (2020); Biloš et al. (2021); Schirmer et al. (2022); Ansari et al. (2023), ContiFormer Chen et al. (2023), and set or graph models Horn et al. (2020); Yalavarthi et al. (2024); Li et al. (2025); Liu et al. (2026a) retain timestamps, masks, time gaps, or irregular graph structure.

Despite these efforts to represent irregular observations, few existing methods explicitly model sampling density. The distribution of observation times within local regions remains largely underutilized, although it provides additional information about the irregular sampling process. D^{2}ARIF explicitly estimates observation-time densities for each variable and patch, encodes the resulting local distributions, and incorporates them into forecasting.

2.2 Cross-Variable Interaction in Irregular Multivariate Time Series

Global interaction models mix variable histories through shared continuous states Rubanova et al. (2019); Schirmer et al. (2022); Ansari et al. (2023), variable graphs Mercatali et al. (2024), hyperedges Li et al. (2025), or compressed frequency representations Zhou et al. (2026). Local methods exchange information at reference times or within corresponding patches Shukla & Marlin (2021); Zhang et al. (2024); Liu et al. (2025); Hu et al. (2025); Luo et al. (2025), while graph methods propagate information through learned neighborhoods Zhang et al. (2022); Yalavarthi et al. (2024). ASTGI Liu et al. (2026a) connects selected non-adjacent observations through sparse spatio-temporal edges. These mechanisms capture broad or local dependencies, but access to a particular source history is mediated by aggregation, bounded regions, or graph paths.

D^{2}ARIF takes a different approach from the methods above: it lets each target retrieve past information from memories that retain the identity of their source variables. To realize this design, D^{2}ARIF builds on the delta rule for memory updates introduced by DeltaNet Schlag et al. (2021) and adapts it to irregular multivariate forecasting in three ways. It maintains a separate state for each source variable, scales decay by the actual elapsed time between active patches, and uses each target variable's

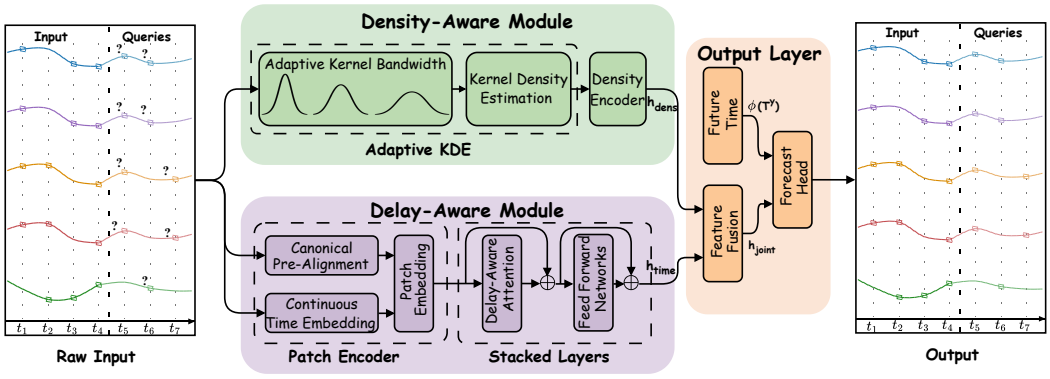


Figure 1: D^{2}ARIF encodes historical observations along two paths. The Delay-Aware Module retains and retrieves asynchronous cross-variable evidence, while the Density-Aware Module represents local observation-time densities. Their representations are fused with future query times for forecasting.

query to select and retrieve information from the source states. These changes preserve source identity, account for irregular observation intervals, and allow different targets to select different source histories.

3 Method

3.1 Overview

Figure 1 presents D^{2}ARIF. Canonical Pre-Alignment (CPA) places observations and availability masks on the union of their timestamps without imputing unobserved values Zhou et al. (2026). The Density-Aware Module estimates local observation-time densities from historical timestamps. In parallel, continuous-time patch encoding and the Delay-Aware Module produce a temporal representation. The output layer fuses both representations with future query times for forecasting.

3.2 Density-Aware Module

3.2.1 Local Density Estimation

For patch i and variable c, we retain the normalized timestamps at which the variable is observed,

$$\mathcal{E}_{i,c} = \{u_{i,l} \mid l \in I_i,\ M_{l,c} = 1\}, \qquad u_{i,l} = \frac{t_l - \min_{q\in I_i} t_q}{\max_{q\in I_i} t_q - \min_{q\in I_i} t_q + \epsilon}. \tag{1}$$

We evaluate density on a fixed grid $\mathcal{G} = \{g_r\}_{r=1}^{G}$ over $[0,1]$, with spacing Δg and G. SSVKernel selects a locally varying bandwidth from $\mathcal{E}_{i,c}$ Shimazaki & Shinomoto (2010). This optimization is performed independently for each variable in each nonempty patch. For every candidate $w \in \mathcal{W}$, we first form the fixed-bandwidth estimate

$$\widetilde{f}_{i,c,w}(g) = \frac{1}{|\mathcal{E}_{i,c}|} \sum_{u\in\mathcal{E}_{i,c}} \frac{1}{\sqrt{2\pi}w} \exp\left[-\frac{(g-u)^2}{2w^2}\right] \tag{2}$$

and estimates its pointwise L_2 risk using the empirical timestamp density $\widehat{p}_{i,c}$,

$$\widetilde{C}_{i,c,w}(g) = \widetilde{f}_{i,c,w}(g)^2 - 2\widetilde{f}_{i,c,w}(g)\widehat{p}_{i,c}(g) + \frac{2\widehat{p}_{i,c}(g)}{\sqrt{2\pi}w}. \tag{3}$$

To make bandwidth selection depend on the local observation pattern, the cost is averaged at multiple neighborhood scales ℓ, and the minimizing candidate is selected at each location,

$$w^{\star}_{i,c,\ell}(g) = \arg\min_{w\in\mathcal{W}} \left(K_\ell * \widetilde{C}_{i,c,w}\right)(g), \tag{4}$$

where K_ℓ is a Gaussian local averaging kernel with scale $\ell \in \mathcal{W}$. SSVKernel then constructs a smooth bandwidth profile $b_{i,c}^{(\gamma)}(g)$ by matching the ratio $w_{i,c,\ell}^{\star}(g)/\ell$ to a stiffness parameter γ and applying Nadaraya–Watson kernel regression across locations. The stiffness is chosen independently for each (i, c) by minimizing the global estimated risk,

$$\gamma_{i,c}^{\star} = \underset{0<\gamma\leq 1}{\arg\min} \sum_{r=1}^{G} \left[\widehat{f}_{i,c}^{(\gamma)}(g_r)^2 - 2\widehat{f}_{i,c}^{(\gamma)}(g_r)\widehat{p}_{i,c}(g_r) + \frac{2\widehat{p}_{i,c}(g_r)}{\sqrt{2\pi} b_{i,c}^{(\gamma)}(g_r)} \right] \Delta g. \tag{5}$$

where $\widehat{f}_{i,c}^{(\gamma)}$ is the density induced by the smoothed profile and Δg is the grid spacing. The one-dimensional search for $\gamma_{i,c}^{\star}$ uses golden-section optimization on $(0, 1]$. We use $b_{i,c,r} = b_{i,c}^{(\gamma_{i,c}^{\star})}(g_r)$ in the final estimate,

$$f_{i,c}(g_r) = \frac{1}{|\mathcal{E}_{i,c}|} \sum_{u\in\mathcal{E}_{i,c}} \frac{1}{\sqrt{2\pi} b_{i,c,r}} \exp\left[-\frac{(g_r - u)^2}{2b_{i,c,r}^2} \right]. \tag{6}$$

Thus, dense regions receive narrower kernels while sparse regions are smoothed more strongly, without learning the bandwidth estimator jointly with the forecaster. An empty patch produces the zero density vector, and a fixed positive bandwidth is used when adaptive estimation is not numerically defined. We normalize each nonempty density vector to unit mass, yielding $\bar{\mathbf{f}}_{i,c} \in \mathbb{R}^G$, and stack all vectors as $\bar{\mathbf{F}} \in \mathbb{R}^{N_p\times C\times G}$.

3.2.2 Density Representation

The Density-Aware Module maps each local distribution to the hidden dimension,

$$\mathbf{h}'_{i,c} = \mathrm{MLP}_{\mathrm{density}}\big(\log(1 + \bar{\mathbf{f}}_{i,c})\big), \qquad \mathbf{H}' \in \mathbb{R}^{N_p\times C\times d}. \tag{7}$$

Let $M'_{i,c}$ indicate whether variable c is observed in patch i. We aggregate only nonempty patches,

$$\mathbf{h}_{\mathrm{density},c} = \frac{\sum_{i=1}^{N_p} M'_{i,c}\mathbf{h}'_{i,c}}{\max\Big(1, \sum_{i=1}^{N_p} M'_{i,c}\Big)}, \qquad \mathbf{H}_{\mathrm{density}} \in \mathbb{R}^{C\times d}. \tag{8}$$

This aggregation preserves a separate density representation for every variable and introduces no additional temporal observations; the module uses historical timestamps only.

3.3 Delay-Aware Module

3.3.1 Patch Encoding

At the patch encoding level, we adopt a fixed number of sampling points p for patch partitioning, yielding $N_p = \lceil \frac{L}{p} \rceil$ patches, where the i-th patch covers the index set I_i. Let $\mathbf{T}_i = \mathbf{T}[I_i] \in \mathbb{R}^p$ denote the timestamps contained within each patch. For variable c, if there is at least one observation within the i-th patch, we set $M'_{i,c} = 1$; otherwise, $M'_{i,c} = 0$. This yields the patch-level mask $\mathbf{M}'$.

For any timestamp t, we employ a continuous time embedding:

$$\phi(t)[i] = \begin{cases} \omega_i t + \alpha_i, & i = 0, \\ \sin(\omega_i t + \alpha_i), & \text{otherwise}, \end{cases} \tag{9}$$

where i indexes the embedding dimension, and both ω_i and α_i are learnable parameters. The linear component captures non-periodic trends, while the remaining sinusoidal components capture periodic variations at different time scales. For each irregular time series, we concatenate the continuous-time embeddings of historical timestamps, observations, and masks, and encode them via an MLP to obtain the patch matrix $\mathbf{H} \in \mathbb{R}^{N_p\times C\times d}$. Specifically, the encoding for the i-th patch of the c-th variable is given by:

$$\mathbf{h}_{i,c} = \mathrm{MLP}\left([\phi(t_l) \,\|\, X_{l,c} \,\|\, M_{l,c}]_{l\in I_i} \right) + \mathrm{PE} \tag{10}$$

where PE denotes a learnable positional encoding.

3.3.2 DELAY-AWARE ATTENTION

The Delay-Aware Module models delayed cross-variable dependencies with persistent source-specific states. Its Delay-Aware Attention block follows the error-correcting delta-rule write of DeltaNet Schlag et al. (2021). D^{2}ARIF organizes this memory by source variable, decays the states according to the elapsed interval between active patches, and uses each target variable's query to weight the source states.

The layer processes patches recurrently in temporal order. For layer k, $\mathbf{S}_{i,j} \in \mathbb{R}^{d\times d}$ is the state of source variable j after patch i, and $\mathbf{H}_i^{(k-1)} \in \mathbb{R}^{C\times d}$ is the input representation. Each layer has independent parameters and zero-initialized states; layer superscripts on $\mathbf{S}$ are omitted for readability.

Normalization and projection of the previous-layer representation yield keys, values, and queries:

$$\bar{\mathbf{H}}_i = \mathrm{LN}(\mathbf{H}_i^{(k-1)}), \qquad \mathbf{Q}_i = \mathrm{norm}(\bar{\mathbf{H}}_i\mathbf{W}_q^\top), \mathbf{K}_i = \mathrm{norm}(\bar{\mathbf{H}}_i\mathbf{W}_k^\top), \quad \mathbf{V}_i = \bar{\mathbf{H}}_i\mathbf{W}_v^\top, \tag{11}$$

where $\mathbf{K}_i, \mathbf{Q}_i \in \mathbb{R}^{C\times d}$, $\mathbf{V}_i \in \mathbb{R}^{C\times d}$, LN denotes layer normalization, and norm denotes row-wise ℓ_2 normalization. $\mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_q$ are learnable projections.

Let τ_i denote the timestamp of the first sampling point in patch i. The elapsed interval $\Delta_i = \max(\tau_i - \tau_{i^-}, 0)$ is shared by all source variables in the sample, with $\Delta_i = 0$ for the first active patch. The current representations produce source- and key-specific nonnegative decay rates $\mathbf{E}_i \in \mathbb{R}_+^{C\times d}$:

$$\mathbf{E}_i = \mathrm{softplus}\big(\bar{\mathbf{H}}_i\mathbf{W}_\eta^\top + \mathbf{1}\mathbf{b}_\eta^\top\big)\,. \tag{12}$$

Here $\mathbf{W}_\eta$ and $\mathbf{b}_\eta$ are learnable, and $\mathbf{1}$ is an all-ones column vector. The source states are stacked as $\mathbf{S}_i \in \mathbb{R}^{C\times d\times d}$.

For the update below, $\mathbf{S}_i^- \in \mathbb{R}^{C\times d\times d}$ denotes the stack of decayed source states, $\mathbf{E}_i \in \mathbb{R}_{\geq 0}^{C\times d}$, and $\Delta_i \in \mathbb{R}_{\geq 0}$. The factor $\exp(-\Delta_i\mathbf{E}_i) \in \mathbb{R}^{C\times d}$ is broadcast over the value dimension. Every source state is decayed before the current write:

$$\mathbf{S}_i^- = \exp(-\Delta_i\mathbf{E}_i) \odot \mathbf{S}_{i^-}, \tag{13}$$

where $\odot$ denotes the broadcast elementwise product. For source j, $\mathbf{k}_{i,j}, \mathbf{v}_{i,j}, \bar{\mathbf{h}}_{i,j} \in \mathbb{R}^d$, $M'_{i,j} \in \{0,1\}$, $\mathbf{w}_\beta \in \mathbb{R}^d$, and $b_\beta \in \mathbb{R}$. The delta-rule write for source j is

$$\mathbf{S}_{i,j} = \mathbf{S}_{i,j}^- + M'_{i,j}\,\sigma(\mathbf{w}_\beta^\top\bar{\mathbf{h}}_{i,j} + b_\beta)\,\mathbf{k}_{i,j}\left(\mathbf{v}_{i,j} - (\mathbf{S}_{i,j}^-)^\top\mathbf{k}_{i,j}\right)^\top. \tag{14}$$

Here $\mathbf{S}_{i,j}, \mathbf{S}_{i,j}^- \in \mathbb{R}^{d\times d}$. The patch mask $M'_{i,j}$ prevents writes from unobserved variables, while their previous states still undergo elapsed-time decay. The sigmoid term controls write strength, and the key-value reconstruction residual gives the outer-product correction.

Let $\mathcal{J}_i = \{j \mid \sum_{q=1}^{i} M'_{q,j} > 0\}$ be the sources observed by patch i. For target variable c and source $j \in \mathcal{J}_i$, with $\mathbf{q}_{i,c} \in \mathbb{R}^d$, $\mathbf{w}_\pi \in \mathbb{R}^d$, and $b_{\pi,c,j} \in \mathbb{R}$, the scalar logit $e_{i,c,j} \in \mathbb{R}$ and normalized weight $\pi_{i,c,j} \in \mathbb{R}$ are computed as

$$e_{i,c,j} = \mathbf{w}_\pi^\top\mathbf{S}_{i,j}^\top\mathbf{q}_{i,c} + b_{\pi,c,j}, \qquad \pi_{i,c,j} = \frac{\exp(e_{i,c,j})}{\sum_{q\in\mathcal{J}_i}\exp(e_{i,c,q})}. \tag{15}$$

The target readout and its stacked matrix are, respectively, $\mathbf{r}_{i,c} \in \mathbb{R}^d$ and $\mathbf{R}_i \in \mathbb{R}^{C\times d}$:

$$\mathbf{r}_{i,c} = \sum_{j\in\mathcal{J}_i}\pi_{i,c,j}\,\mathbf{q}_{i,c}^\top\mathbf{S}_{i,j}, \qquad \mathbf{R}_i = [\mathbf{r}_{i,1}; \ldots; \mathbf{r}_{i,C}] \in \mathbb{R}^{C\times d}. \tag{16}$$

For each active patch, after $\mathbf{Q}_i$, $\mathbf{K}_i$, $\mathbf{V}_i$, and $\mathbf{E}_i$ have been computed, the state update order is: (i) decay all source states; (ii) apply the delta-rule write to every observed source; (iii) compute target-specific readout weights from the updated states; and (iv) form the target readouts and the residual

layer output. All source variables are updated synchronously: the writes for every source use the same pre-update state $\mathbf{S}_i^-$, so no source can read another source's newly written state within the same patch. The operations for all variables and patch entries are otherwise carried out in parallel.

Memory attention maps $\mathbf{R}_i \in \mathbb{R}^{C\times d}$ back to the hidden dimension and produces $\mathbf{H}_i^{(k)} \in \mathbb{R}^{C\times d}$ via a residual connection:

$$\mathbf{H}_i^{(k)} = \mathrm{LN}\Big(\mathbf{H}_i^{(k-1)} + \mathrm{MLP}(\mathbf{R}_i)\Big). \tag{17}$$

3.4 Output Layer

3.4.1 Feature Fusion and Forecasting

The Delay-Aware Module output at the final patch gives $\mathbf{H}_{\text{time}} = \mathbf{H}_{N_p}^{(K)} \in \mathbb{R}^{C\times d}$, matching the density representation $\mathbf{H}_{\text{density}}$. For future query q of target variable c, we concatenate both variable-specific representations with the query-time embedding:

$$\mathbf{H}_{\text{joint}} = [\mathbf{H}_{\text{time}} \| \mathbf{H}_{\text{density}} \| \phi(T^y)], \qquad \hat{Y}_{\text{joint}} = \mathrm{MLP}(\mathbf{H}_{\text{joint}}). \tag{18}$$

Stacking predictions over queries and variables yields the final forecast $\hat{\mathbf{Y}}_{\text{joint}}$.

3.4.2 Training Objective

The model is trained on the observed future target entries using MAE:

$$\mathcal{L} = \mathrm{MAE}\Big(\hat{\mathbf{Y}}_{\text{joint}}, \mathbf{Y}\Big). \tag{19}$$

4 Experiments

4.1 Datasets and Metrics

We evaluate D^{2}ARIF on four IMTS forecasting benchmarks: PhysioNet, MIMIC, HumanActivity, and USHCN. They cover clinical measurements, human activity, and climate observations with different variable counts and sampling patterns. We report mean squared error (MSE) and mean absolute error (MAE), with lower values indicating better forecasts.

4.2 Baselines

We compare D^{2}ARIF with 16 baselines. The recurrent and continuous-time group includes GRU-D Che et al. (2018), NeuralFlows Biloš et al. (2021), CRU Schirmer et al. (2022), and GNeuralFlow Mercatali et al. (2024). The set, alignment, and attention group includes PrimeNet Chowdhury et al. (2023), SeFT Horn et al. (2020), mTAN Shukla & Marlin (2021), and Warpformer Zhang et al. (2023). Graph and patch methods include Raindrop Zhang et al. (2022), tPatchGNN Zhang et al. (2024), GraFITi Yalavarthi et al. (2024), Hi-Patch Luo et al. (2025), APN Liu et al. (2026b), and ASTGI Liu et al. (2026a). We also include the recent forecasting designs KAFNet Zhou et al. (2026) and TFMixer Qiu et al. (2026).

4.3 Main Results

We first evaluate D^{2}ARIF against existing IMTS forecasting methods on four real-world benchmarks. Table 1 compares D^{2}ARIF with all 16 baselines using MSE and MAE. ASTGI is included as a close comparison for delayed cross-variable interaction, with both metrics evaluated under the same data and evaluation protocol.

D^{2}ARIF achieves the lowest MAE on all four datasets and the lowest MSE on HumanActivity and USHCN. Warpformer and TFMixer remain slightly better in MSE on PhysioNet and MIMIC, respectively, so D^{2}ARIF ranks second on those two MSE comparisons. ASTGI is consistently

Table 1: Forecasting performance on four IMTS datasets. Overall performance is evaluated by MSE and MAE ($\times 10^{-2}$). The best and second-best results are highlighted in bold and with an underline.

Model	HumanActivity		USHCN		PhysioNet		MIMIC	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
PrimeNet	425.1±0.41	170.2±0.11	49.30±0.15	49.54±0.18	79.53±0.00	68.59±0.01	90.73±0.01	66.14±0.01
NeuralFlows	17.22±0.90	31.50±0.94	20.87±2.58	31.57±1.87	40.56±0.33	44.66±0.27	60.85±1.01	53.06±0.66
CRU	13.87±0.73	26.07±0.92	21.68±1.62	31.80±2.48	61.79±0.45	57.78±0.31	58.95±0.92	51.51±0.48
mTAN	9.930±0.26	22.19±0.47	55.61±20.2	50.15±9.68	38.09±0.43	42.91±0.35	94.08±11.3	67.55±4.59
SeFT	137.9±0.24	97.62±0.07	33.45±0.22	40.83±0.84	77.21±0.21	67.60±0.29	92.30±0.15	66.28±0.08
GNeuralFlow	39.36±15.9	45.41±8.41	22.05±4.21	32.86±4.12	82.07±3.10	67.59±1.00	89.57±2.09	64.50±0.72
GRU-D	18.93±6.27	32.53±4.85	20.97±4.93	30.45±3.05	34.19±0.29	39.92±0.11	47.59±1.00	45.26±0.55
Raindrop	9.161±0.72	21.14±0.72	20.35±3.36	30.29±2.64	34.78±0.19	40.44±0.20	67.54±18.3	54.44±8.68
Warpformer	4.490±0.10	12.28±0.18	18.88±5.98	29.39±5.91	**30.56±0.11**	36.61±0.16	43.02±0.35	40.25±0.14
tPatchGNN	4.430±0.09	12.47±0.31	18.85±4.03	30.84±4.79	31.33±0.53	36.97±0.49	44.31±1.15	40.77±0.88
GraFITi	4.370±0.05	12.21±0.17	16.91±0.93	27.77±2.48	31.75±0.15	36.37±0.36	43.59±4.55	41.42±2.97
Hi-Patch	4.350±0.02	12.04±0.09	17.49±2.68	27.17±2.16	30.71±0.29	36.75±0.42	42.79±0.10	40.33±0.32
KAFNet	4.290±0.03	11.61±0.10	16.98±1.81	26.90±2.26	31.64±0.28	37.15±0.38	44.02±0.86	41.02±0.41
APN	4.210±0.01	11.59±0.06	15.90±1.37	26.11±1.67	30.93±0.11	36.50±0.26	42.92±0.27	40.16±0.16
ASTGI	4.220±0.04	11.62±0.06	16.80±1.52	27.04±0.94	31.04±0.24	35.99±0.37	44.60±0.46	40.60±0.34
TFMixer	4.153±0.04	11.24±0.05	15.18±0.51	21.39±0.36	31.18±0.23	35.76±0.19	**41.82±0.37**	38.64±0.53
D^{2}ARIF	**4.142±0.04**	**11.10±0.06**	**14.15±0.57**	**20.97±0.41**	30.69±0.09	**35.30±0.09**	42.50±0.28	**38.61±0.69**

above D^{2}ARIF on both MSE and MAE across all four datasets, providing a direct comparison with an adaptive spatio-temporal graph interaction mechanism. Overall, D^{2}ARIF ranks first on six of the eight reported benchmark metrics. The clearest MSE gain over the strongest competing baseline occurs on USHCN, where D^{2}ARIF improves over TFMixer by 6.8%.

4.4 Ablation Study

To verify whether the proposed module offers benefits beyond standard attention and whether local observation-time information provides complementary signals, we conduct two comparisons on four real-world benchmarks. First, we replace the entire Delay-Aware Module with a temporal self-attention module that preserves the same output dimensionality, while keeping the density branch, patch encoder, decoder, and forecasting interface unchanged. Second, we remove only the Density-Aware Module while retaining the complete Delay-Aware Module.

Table 2: Main module ablations in MSE and MAE ($\times 10^{-2}$). In w/o Delay-Aware Module, the complete Delay-Aware Module is replaced by a shape-matched temporal self attention module. In w/o Density-Aware Module, only the density branch is removed.

Variant	HumanActivity		USHCN		PhysioNet		MIMIC	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
w/o Delay-Aware Module	4.573	12.36	14.78	21.52	34.15	38.42	46.84	41.57
w/o Density-Aware Module	4.415	11.65	16.33	24.18	31.98	36.55	43.77	39.28
D^{2}ARIF	4.142	11.10	14.15	20.97	30.69	35.30	42.50	38.61

Table 2 shows a consistent separation between the full model and both replacements. Replacing the Delay-Aware Module with temporal self-attention increases every reported error, with an average degradation of 8.27% in MSE and 7.00% in MAE. The larger gaps on HumanActivity and PhysioNet suggest that explicit source-specific memory and elapsed-time retrieval are most useful when asynchronous cross-variable dependencies cannot be summarized by content similarity alone; the smaller, but still consistent, gains on USHCN indicate that the same mechanism remains beneficial when density and sampling regularity provide stronger auxiliary cues. Removing Density-Aware Module also worsens all eight metrics, with average increases of 6.62% in MSE and 5.78% in MAE. Its largest effect is on USHCN, where the MSE and MAE gaps exceed 13%, consistent with the stronger value of observation-time structure in a dataset whose sampling patterns vary substantially across variables. Together, these comparisons indicate that the two branches contribute complementary information: the temporal branch resolves delayed interactions, while Density-Aware Module captures sampling structure that is unavailable to the value path alone.

4.5 GENERALITY ACROSS BACKBONES

We evaluate whether the Density-Aware Module transfers to different forecasting backbones. We insert it immediately before the decoder of PrimeNet, GRU-D, Raindrop, GraFITi, and TFMixer while retaining the rest of each model and its training configuration. Table 3 compares the original and augmented versions on four datasets.

Table 3: DensAM generality across five backbones. DensAM denotes the Density-Aware Module, and +DensAM denotes the backbone augmented with this module. Values are MSE ($\times 10^{-2}$).

Model	Variant	HumanActivity	USHCN	PhysioNet	MIMIC	Improvement
PrimeNet	Original	425.1±0.41	49.30±0.15	79.53±0.00	90.73±0.01	–
	+DensAM	**419.2±0.48**	**35.49±0.27**	**75.39±0.16**	**85.29±0.25**	**10.15%**
GRU-D	Original	18.93±6.27	20.97±4.93	34.19±0.29	47.59±1.00	–
	+DensAM	**18.72±2.04**	**17.09±2.01**	**32.48±0.24**	**44.64±0.68**	**7.70%**
Raindrop	Original	9.161±0.72	20.35±3.36	34.78±0.19	67.54±18.3	–
	+DensAM	**9.088±0.44**	**18.31±1.97**	**33.08±0.16**	**63.42±8.73**	**5.45%**
GraFITi	Original	4.370±0.05	16.91±0.93	31.75±0.15	43.59±4.55	–
	+DensAM	**4.344±0.05**	**15.22±0.68**	**30.87±0.12**	**42.84±2.19**	**3.77%**
TFMixer	Original	4.153±0.04	15.18±0.51	31.18±0.23	41.82±0.37	–
	+DensAM	**4.144±0.04**	**14.26±0.45**	**31.13±0.19**	**41.71±0.32**	**1.68%**
Improvement		**0.82%**	**14.52%**	**3.61%**	**4.06%**	**5.75%**

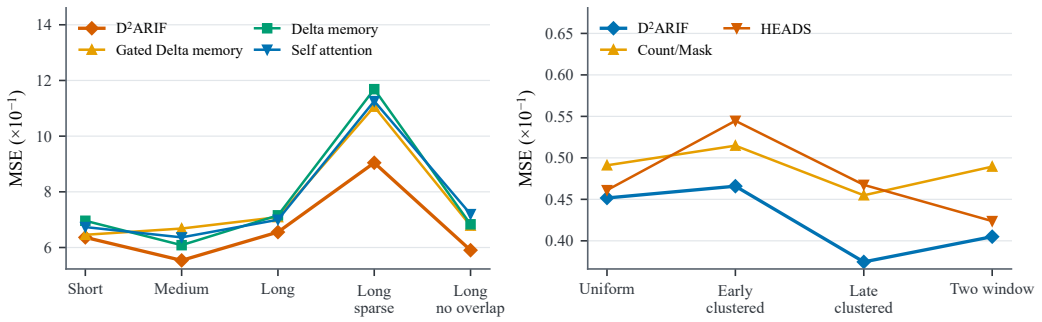
Adding the Density-Aware Module lowers MSE in all 20 evaluations, with a mean relative reduction of 5.75%. The reductions are largest on USHCN, and PrimeNet and GRU-D benefit more than the other backbones. HumanActivity and TFMixer show smaller gains, while the direction of the change remains consistent. The Density-Aware Module transfers across temporal backbones, with the magnitude of the gain determined by both the model and the dataset.

4.6 SCOPE OF APPLICABILITY

We conduct two controlled experiments to characterize the conditions under which the two modules contribute to forecasting performance. The first varies the delay and availability of historical observations to examine when delayed cross-variable information matters. The second keeps observation counts fixed while varying the distribution of timestamps within each historical patch to examine when local observation-time structure matters. The generation procedure, split construction, and training protocol are given in Appendix A.

For the delay experiment, across delay conditions we share the same source-process trajectories and random innovations, and vary only the delay parameter τ. The target process y is regenerated for each τ and therefore changes with the delay. Models are trained and evaluated separately for each delay condition, so the experiment measures modeling performance under different delay conditions rather than generalization to unseen delays or an observation-delay intervention on a fixed target trajectory. The long-sparse and long-asynchronous settings retain the long lag while reducing the available history or removing timestamp overlap, respectively. For the local distribution experiment, each sample is generated from a continuous four-variable process and belongs to one of four timestamp patterns. Every variable has eight observations in each of six patches, so the total count and every patch count remain fixed; only the locations of those observations within each patch change. The patterns distribute observations uniformly, concentrate them near the beginning or end, or split them between the two ends. They are mixed into one balanced training dataset, while test scores are reported separately by pattern. We compare the complete Density-Aware Module with Count-and-Mask Representation and HEADS Hu et al. (2026). Count and Mask tests whether patch counts suffice, and HEADS tests whether a summary over the entire window can capture the distribution within each patch.

In Figure 2(a), the Delay-Aware Module's complete MSE values across the five lag and availability conditions are reported in Appendix Table 6. This result characterizes delayed retrieval under histories that are farther away, less available, or not temporally aligned. Figure 2(b) shows the test MSE



(a) Lag from source to target (b) Local timestamp distribution

Figure 2: Controlled synthetic experiments for the applicability of the two modules. Panel (a) reports MSE across lag conditions, including long histories with sparse observations and no timestamp overlap between the relevant variables. Panel (b) fixes the global and per-patch observation counts, varies the timestamp distribution within each patch.

of the density representations; the complete values are reported in Appendix Table 7. HEADS is close to D^{2}ARIF for the uniform pattern, whereas the gap is larger for clustered timestamp patterns. These findings suggest that our proposed moudel provide additional predictive information. Importantly, the extent to which these temporal patterns are present varies across real-world datasets, leading to differences in the practical benefits of the two modules. In our experiments, USHCN exhibits a more pronounced performance improvement, while the gains on the other datasets are more modest.

4.7 Parameter Sensitivity

Figure 3 shows that the impact of the model dimension d varies significantly with the dataset scale. Smaller datasets, such as Human Activity and USHCN, tend to perform better with lower dimensions; excessive dimensionality in these cases often leads to performance degradation due to overfitting. Conversely, larger datasets like PhysioNet and MIMIC require higher dimensions to provide sufficient representational capacity for data fitting.

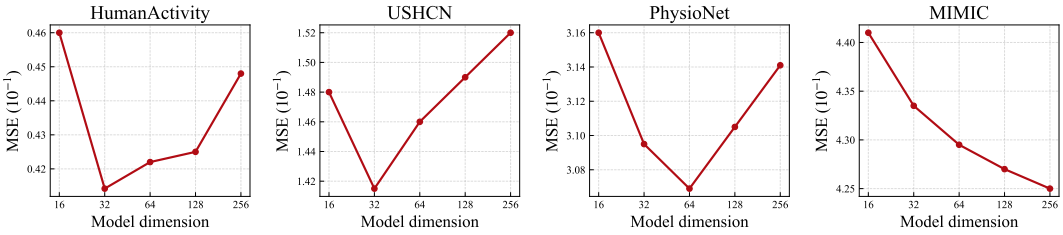


Figure 3: Sensitivity to hidden dimension d on four datasets.

5 Conclusion

We presented D^{2}ARIF, which combines delayed cross-variable interaction with local observation-time density encoding for IMTS forecasting. On four benchmarks, replacing the Delay-Aware Module with temporal self-attention raises both MSE and MAE, while removing Density-Aware Module also worsens all eight metrics; these comparisons support the contributions of both components in the full model. Across the main benchmark comparison, D^{2}ARIF ranks first on six of eight metrics. The controlled experiments further show that delayed retrieval remains competitive when information comes from earlier observations, including sparse observations and observations recorded at different times. The density study holds static count statistics fixed and isolates predictive information in local timestamp distributions.

Ethics statement

This study uses established IMTS benchmarks and introduces no new human-subject data.

AI Usage Statement

In this work, we used generative AI tools for research execution only, not for research ideation, and for drafting sections of the paper. We have not used generative AI tools for generating synthetic datasets or proving mathematical claims, and no other required disclosure tasks are applicable to this work. Additionally, we used generative AI tools to aid or polish writing and for retrieval and discovery, such as finding related work. We have reviewed all AI-assisted work. AI-assisted writing and paper drafts were checked by the authors for accuracy, originality, and potential plagiarism through a manual literature survey. AI-assisted retrieval results were verified by manual inspection. AI-assisted research execution was reviewed and validated by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI. Details are described in the paper.

Reproducibility statement

Appendix A gives the complete generation and evaluation protocol for the controlled experiments. Tables report means and standard deviations where available, and all external datasets and baselines are identified by citation. The complete codebase will be open-sourced upon paper acceptance.

References

Abdul Fatir Ansari, Alvin Heng, Andre Lim, and Harold Soh. Neural continuous-discrete state space models for irregularly-sampled time series. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 926–951. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/ansari23a.html.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural odes. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 21325–21337. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b21f9f98829dea9a48fd8aaddc1f159d-Paper.pdf.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8 (1):6085, 2018. doi: 10.1038/s41598-018-24271-9. URL https://doi.org/10.1038/s41598-018-24271-9.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. Contiformer: Continuous-time transformer for irregular time series modeling. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 47143–47175. Curran Associates, Inc., 2023. doi: 10.52202/075280-2042. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/9328208f88ec69420031647e6ff97727-Paper-Conference.pdf.

Ranak Roy Chowdhury, Jiacheng Li, Xiyuan Zhang, Dezhi Hong, Rajesh K. Gupta, and Jingbo Shang. Primenet: Pre-training for irregular multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7184–7192, 2023. doi: 10.1609/aaai.v37i6.25876.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. 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. 4353–4363. PMLR, 13–18 Jul 2020.

Yimian Hu, Jianping Zhou, Bin Lu, Guanjie Zheng, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. Heads: Temporal observation density boosts irregular multivariate time series forecasting. In *Database Systems for Advanced Applications*, pp. 219–236, Singapore, 2026. Springer Nature Singapore. ISBN 978-981-92-0372-7.

Zhangyi Hu, Jiemin Wu, Hua Xu, Mingqian Liao, Ninghui Feng, Bo Gao, Songning Lai, and Yutao Yue. IMTS is worth time $\times$ channel patches: Visual masked autoencoders for irregular multivariate time series prediction. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24805–24822. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/hu25y.html.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6696–6707. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4a5876b450b45371f6cfe5047ac8cd45-Paper.pdf.

Boyuan Li, Yicheng Luo, Zhen Liu, Junhao Zheng, Jianming Lv, and Qianli Ma. HyperIMTS: Hypergraph neural network for irregular multivariate time series forecasting. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35502–35518. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/li25bl.html.

Jiexi Liu, Meng Cao, and Songcan Chen. Timecheat: A channel harmony strategy for irregularly sampled multivariate time series analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18861–18869, 2025. doi: 10.1609/aaai.v39i18.34076.

Xvyuan Liu, Xiangfei Qiu, Hanyin Cheng, Xingjian Wu, Chenjuan Guo, Bin Yang, and Jilin Hu. ASTGI: Adaptive spatio-temporal graph interactions for irregular multivariate time series forecasting. In *International Conference on Learning Representations*, 2026a.

Xvyuan Liu, Xiangfei Qiu, Xingjian Wu, Zhengyu Li, Chenjuan Guo, Jilin Hu, and Bin Yang. Rethinking irregular time series forecasting: A simple yet effective baseline. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23873–23881, 2026b. doi: 10.1609/aaai.v40i28.39563.

Yicheng Luo, Bowen Zhang, Zhen Liu, and Qianli Ma. Hi-patch: Hierarchical patch GNN for irregular multivariate time series. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41494–41519. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/luo25r.html.

Giangiacomo Mercatali, Andre Freitas, and Jie Chen. Graph neural flows for unveiling systemic interactions among irregularly sampled time series. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 57183–57206. Curran Associates, Inc., 2024. doi: 10.52202/079017-1822. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/68b8d2bc77268facfc75a78782da9559-Paper-Conference.pdf.

Xiangfei Qiu, Kangjia Yan, Xvyuan Liu, Xingjian Wu, and Jilin Hu. Bridging time and frequency: A joint modeling framework for irregular multivariate time series forecasting. In *ICML*, 2026.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. 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/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19388–19405. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/schirmer22a.html.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. 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. 9355–9366. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

Hideaki Shimazaki and Shigeru Shinomoto. Kernel bandwidth optimization in spike rate estimation. *Journal of Computational Neuroscience*, 29(1–2):171–182, 2010. doi: 10.1007/s10827-009-0180-4.

Satya Narayan Shukla and Benjamin Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=4c0J6lwQ4_.

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

Vijaya Krishna Yalavarthi, Kiran Madhusudhanan, Randolf Scholz, Nourhan Ahmed, Johannes Burchert, Shayan Jawed, Stefan Born, and Lars Schmidt-Thieme. Grafiti: Graphs for forecasting irregularly sampled time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16255–16263, 2024. doi: 10.1609/aaai.v38i15.29560.

Jiawen Zhang, Shun Zheng, Wei Cao, Jiang Bian, and Jia Li. Warpformer: A multi-scale modeling approach for irregular clinical time series. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 3273–3285, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599543. URL https://doi.org/10.1145/3580305.3599543.

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular multivariate time series forecasting: A transformable patching graph neural networks approach. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60179–60196. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhang24bw.html.

Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. Graph-guided network for irregularly sampled multivariate time series. In *International Conference on Learning Representations, ICLR*, 2022.

Ziyu Zhou, Yiming Huang, Yanyun Wang, Yuankai Wu, James Kwok, and Yuxuan Liang. Revitalizing canonical pre-alignment for irregular multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29115–29123, 2026. doi: 10.1609/aaai.v40i34.40149.

A CONTROLLED SYNTHETIC EXPERIMENT PROTOCOL

The synthetic experiments are designed as controlled conditional capability tests. Their purpose is to characterize the conditions under which each proposed module can contribute to forecasting. SyntheticLag constructs a forecasting process with an explicitly specified delayed source-to-target dependency. By varying the delay length, source sparsity, and source-target timestamp overlap, it evaluates whether the Delay-Aware Module can exploit source-specific information when the corresponding dependency is present under different observation patterns. SyntheticDensity keeps the global and per-patch observation counts fixed while changing the within-patch timestamp layout. It evaluates whether an explicit representation of local timestamp distributions can improve forecasting under this controlled sampling setup beyond count and mask statistics.

A.1 SHARED DATA AND EVALUATION CONVENTIONS

Each trajectory has $C = 4$ variables, $L = 96$ historical grid positions, and $H = 24$ future query positions. Historical indices are $n = 0, \ldots, 95$ with normalized times $t_n = n/95$, and future queries are $n = 96, \ldots, 119$. We use patches of $p = 16$ grid positions, producing six historical patches. Dataset splits are made by latent trajectory before observation masks or views with different timestamp patterns are constructed. The training, validation, and test splits contain 2,048, 256, and 256 latent trajectories, respectively. Table 4 summarizes these dimensions and split sizes together with the remaining shared settings.

Random streams are independent across splits and trajectories. Each variable is standardized with the mean and standard deviation computed from observed values in the training histories. The same statistics are used for validation and test inputs and are inverted before reporting prediction error.

Table 4: Common configuration for the controlled synthetic experiments.

Item	Setting
Variables	4; one source, one target, and two nuisance variables
History and forecast	96 historical positions and 24 future queries
Patch construction	6 nonoverlapping patches of 16 positions
Train/validation/test	2,048 / 256 / 256 latent trajectories
Generator	NumPy PCG64 with a `SeedSequence` for each trajectory
Selection and reporting	Validation MAE; test MSE after inverse scaling

A.2 DELAYED CROSS-VARIABLE RETRIEVAL

Latent process. For every trajectory, the source s_n and nuisance variables $u_n^{(1)}$ and $u_n^{(2)}$ are independent stationary AR(1) processes,

$$z_n = 0.8 z_{n-1} + \sqrt{1 - 0.8^2}\,\epsilon_n, \qquad \epsilon_n \sim \mathcal{N}(0, 1), \tag{20}$$

where z denotes any of the three processes. The target is

$$y_n = 0.5 y_{n-1} + 0.8 s_{n-\tau} + 0.1 \epsilon_n^{(y)}, \qquad \epsilon_n^{(y)} \sim \mathcal{N}(0, 1). \tag{21}$$

To cover the largest lag during warm-up, we initialize each source and nuisance process at $n = -328$ from $\mathcal{N}(0, 1)$ and generate it through $n = 119$. We set $y_{-257} = 0$, generate the target from $n = -256$ through 119, and discard every position before $n = 0$. The innovations and the source and nuisance trajectories are shared across the five conditions of the same trajectory. Only τ and the observation masks for the source and target change. The prediction target is $(y_{96}, \ldots, y_{119})$. Because $\tau \geq 24$, every causal source value for the forecast lies in the historical window.

Observation conditions. For each trajectory, we sample a set A of 48 distinct historical indices uniformly without replacement. The short-, medium-, and long-lag conditions observe both source and target at A, which holds the observation process fixed while changing only τ. The condition with sparse source observations samples a 24-element subset of A for the source and retains A for the target. The asynchronous condition observes the source at A and the target at its complement,

so each has 48 observations but they share no timestamp. Each nuisance variable is observed at an independent set of 48 uniformly sampled historical indices. Values are recorded at the selected normalized grid times without interpolation or imputation.

Table 5: Conditions in SyntheticLag. Observation counts refer to the 96-position history. The final column is the number of timestamps shared by the source and target.

Condition	Lag τ	Source count	Target count	Overlap
Short	72	48	48	48
Medium	84	48	48	48
Long	96	48	48	48
Long + sparse	96	24	48	24
Long + asynchronous	96	48	48	0

Each delay variant is trained separately on each condition; the reported result contains one test result per model-condition pair. The reported delay variants share the patch encoder, decoder, hidden width, depth, optimizer, batch order, early-stopping rule, and training examples. Each replacement changes only the operation stated in its label. Prediction error is computed over the 24 future target values.

A.3 Local Timestamp Distributions

For each sample, we draw independent Fourier coefficients and define a continuous four-variable process with a shared component, channel-specific components, and a channel-specific linear trend. The future targets are the continuation of that sample's process after the historical interval, so the history and prediction target remain linked through the same trajectory. The history interval is divided into six patches. We hold the static count statistics fixed: every variable has exactly eight observations in every patch, giving 48 observations per variable and 192 observations per sample. We then assign the sample one of four local timestamp distributions within each patch: Uniform uses $\mathrm{Beta}(1,1)$, Early-clustered uses $\mathrm{Beta}(2,5)$, Late-clustered uses $\mathrm{Beta}(5,2)$, and Two-window uses an equal mixture of $\mathrm{Beta}(2,8)$ and $\mathrm{Beta}(8,2)$. Thus the global count and every per-patch count are fixed, while the within-patch timestamp distribution changes how the continuous trajectory is observed before forecasting.

The density comparison uses the complete Density-Aware Module, Count-and-Mask Representation, and HEADS (the window-level density representation). All variants use the same temporal backbone, decoder, training split, and optimization protocol. Samples from the four timestamp distributions are mixed in a balanced schedule during training, so each model is trained on one combined dataset rather than on a separate distribution-specific task. Test scores are then stratified by timestamp distribution. This experiment tests whether information about the distribution of observations within each historical patch improves forecasting beyond static count statistics.

A.4 Implementation Details

All experiments were conducted on a server equipped with two NVIDIA RTX 4090 GPUs and an Intel Core i9-13900KF CPU. Across all experiments, the optimizer hyperparameters were fixed to $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$. The main experiments used AdamW with a learning rate of 1×10^{-3} and a weight decay of 0.01, and were trained for at most 200 epochs. Full optimization settings, dataset-specific windows, and reproducibility details are provided in the appendix. The controlled experiments used a hidden width of $d = 64$, three representation layers, and a patch size of $p = 16$, and were trained with Adam using a batch size of 128, gradient-norm clipping at 1, and a maximum of 100 epochs. The learning rate and weight decay were selected from validation grids of $\{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$ and $\{0, 10^{-5}\}$, respectively; the candidate with the lowest validation MAE was selected, and its test checkpoint was evaluated. Training minimized MAE and was stopped after 20 epochs without improvement in validation MAE. Standard deviations and significance claims are not reported for the controlled experiments, and no test results were used for hyperparameter selection.

Generation checks. Before training, we require disjoint identifiers of latent trajectories across the three splits and finite timestamps and values. Automated checks enforce the source count, target count, and overlap in Table 5 and verify that every source index required by a future target lies in the history. Normalization statistics and all validation decisions are computed without test examples.

B Additional Experimental Results

Table 6: Delay-aware mechanism results on the controlled lag task. Values are test MSE ($\times 10^{-1}$), using the same scale as Figure 2(a).

Variant	Short	Medium	Long	Long + sparse	Long + async.
D2ARIF	6.363	5.537	6.554	9.044	5.903
Gated Delta memory	6.457	6.681	7.091	11.058	6.785
Delta memory	6.961	6.084	7.156	11.687	6.834
Self-Attention	6.734	6.364	6.997	11.258	7.199

Table 7: Density mechanism results with fixed global and per-patch counts. Values are test MSE ($\times 10^{-1}$), using the same scale as Figure 2(b). Rows and labels match the plotted methods; the bimodal condition corresponds to the Two-window pattern.

Variant	Uniform	Early-clust.	Late-clust.	Two-window
D2ARIF	0.452	0.466	0.375	0.405
HEADS	0.461	0.545	0.467	0.424
Count-and-Mask Representation	0.491	0.515	0.455	0.489

Variant definitions. In the reported self-attention variant, the complete Delay-Aware Module is replaced by a shape-matched temporal self-attention module. In Count-and-Mask Representation, the local density vector is replaced by observation counts and nonempty masks. In HEADS, the patch-level timestamp representation is replaced by explicitly defined historical and forecast-window density statistics.