DYNAMIC CHANNEL-ENTITY BIPARTITE GRAPH FOR IRREGULAR MULTI-CHANNEL TIME SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Irregularly sampled multi-channel time series exhibit asynchronous observations, sparsity, and uneven sampling intervals, while their sampling patterns may themselves convey task-relevant information. Existing graph-based methods organize interactions by learning channel dependencies over fully connected candidate graphs, constructing channel relations from semantic priors, or building channel-time graphs within predefined temporal patches. However, fixed patching may leave some patches with insufficient information and split key temporal dynamics across patch boundaries, while dense or prior-based graphs may propagate irrelevant information when relations are insufficiently filtered or priors are mismatched. To address these challenges, we propose the Reference-Entity-Augmented Dynamic Graph model (REA-DG), which organizes observation-driven local interactions through a sample entity. REA-DG constructs an entity-channel bipartite graph in which the sample entity connects only to currently observed channels and mediates their interactions. To complement sparse local information, shared learnable reference entities connect to all channels through channel-specific learnable edges, injecting context learned from the training data into message passing. Experiments demonstrate that REA-DG achieves state-of-the-art performance across financial forecasting, clinical time-series classification, and network-traffic forecasting. Code will be available after acceptance.

1 INTRODUCTION

Irregularly sampled multi-variate time series (ISMTS) are common in healthcare (Goldberger et al., 2000; Reyna et al., 2020), finance (Engle & Russell, 1998), and network monitoring (Koumar & Čejka, 2023; Chang et al., 2025), where observations are produced by clinical decisions, market events, or network activity rather than by a synchronized sampling schedule. Channels are observed at different rates, consecutive observations are separated by uneven intervals, and only a subset of channels may be available at a given step. The timing and availability of observations may themselves carry task-relevant information: an ordered test reflects clinical suspicion, a burst of trades or news reflects market attention, and the presence of a flow record reflects network activity. The observation pattern is thus part of the evidence a model must interpret. As the set of observed channels changes, the model must determine how their current values inform one another and how this interaction connects to the sample's history.

Existing approaches differ in whether and how channels exchange information during encoding (fig. 1). Channel-independent (CI) encoders process channels separately, potentially with shared parameters. Although effective in regularly sampled forecasting (Zeng et al., 2023; Nie et al., 2023), independent encoding cannot use other channels to contextualize sparse observations. Channel-dependent (CD) approaches enable such exchange through joint representations or explicit graphs. Non-graph methods combine measurements through recurrent states, continuous-time dynamics, or attention and set aggregation (Che et al., 2018; Rubanova et al., 2019; Shukla & Marlin, 2021; Horn et al., 2020). Graph-based methods instead specify the objects that interact: channel-channel (C-C) graphs represent dependencies between channels (Zhang et al., 2022; 2024; Hajisafi et al., 2025), whereas channel-time (C-T) graphs connect channels to timestamps and encode observed values on edges (Yalavarthi et al., 2024; Liu et al., 2025).

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.

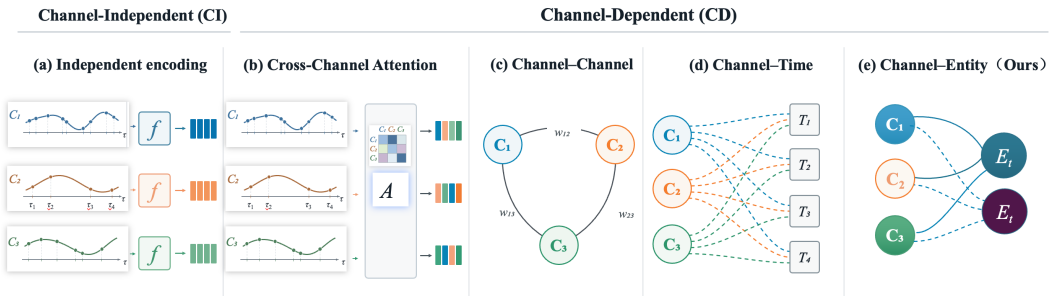

Figure 1: Comparison of channel interaction strategies for multivariate time series. (a) Independent encoding processes each channel separately without cross-channel interaction. (b) Cross-channel attention learns attention weights to aggregate information across channels. (c) Channel–channel graph represent channels as nodes and their dependencies as edges for message passing. (d) Channel–time graphs connect channel nodes to timestamp nodes for within-patch graph interactions (e) Our entity–channel graph connects the sample entity to currently observed channels to mediate observation-driven interactions.

C-C graphs organize cross-channel interaction through channel dependencies, but inferring these dependencies solely from misaligned and imbalanced observations remains challenging in ISMTS (Luo et al., 2024). C-T graphs instead encode measurements on channel-time edges; TimeCHEAT constructs such graphs within fixed temporal patches before applying channel-independent encoding across patches (Liu et al., 2025). However, fixed patching can leave sparse patches with insufficient information for feature extraction and split important temporal dynamics across patch boundaries (Liu et al., 2026b). More broadly, shared latent structures across samples can provide complementary information when individual sequences are sparsely observed (Wu et al., 2021). These challenges motivate an interaction mechanism whose connectivity follows available observations and whose current updates incorporate context accumulated across observation steps.

To address these challenges, we propose Reference-Entity-Augmented Dynamic Graph (REA-DG). As illustrated in fig. 1(e), REA-DG organizes cross-channel interaction through a channel-entity (C-E) graph that connects a sample entity directly to currently observed channels. It processes observations step by step without fixed temporal patch boundaries, using channel memories to incorporate sample-specific history into the entity before current interactions. Observed channels update the entity and receive its contextual feedback, while shared learnable reference entities provide complementary context learned across training samples. This design integrates observation-defined connectivity, historical context, and shared knowledge within the interaction process.

Our main contributions are as follows:

- We propose an observation-driven channel-entity (C-E) graph for irregular time-series modeling. Actual observations determine the connections between nodes, enabling the model to construct meaningful relations without relying on assumed connectivity.
- We introduce shared learnable reference entities alongside real sample entities in C-E graphs for irregular time-series modeling. These reference entities enrich sample-level states with shared knowledge learned from training data, complementing the limited local evidence available from individual observations under extreme sparsity.
- We evaluate REA-DG on clinical time-series classification, financial forecasting, and network-traffic forecasting benchmarks. Our method achieves state-of-the-art performance across these domains and tasks.

2 Related Work

Irregularly sampled multi-variate time series modeling. Existing methods organize temporal modeling and cross-channel interaction at different granularities. GRU-D, ODE-RNN, DGM^2-O, Neural Flows, and CRU incorporate elapsed time into state updates through decay or continuous-time dynamics (Che et al., 2018; Rubanova et al., 2019; Wu et al., 2021; Biloš et al., 2021; Schirmer

et al., 2022). IP-Net and mTAND align observations to reference timestamps through interpolation and temporal attention, respectively (Shukla & Marlin, 2019; 2021). SeFT, STraTS, and DuETT aggregate measurement information across channels and time using set-based or attention-based encoders (Horn et al., 2020; Tipirneni & Reddy, 2022; Labach et al., 2023). Warpformer adaptively resamples irregular representations across temporal scales, while ViTST encodes time series as images (Zhang et al., 2023; Li et al., 2023). Recent patch-based methods explicitly organize temporal processing and cross-channel interaction within and across patches. Among them, t-PatchGNN and Hi-Patch capture temporal and inter-variable dependencies through time-varying channel graphs and hierarchical graph aggregation, respectively (Zhang et al., 2024; Luo et al., 2025), while TimeCHEAT combines channel-dependent processing within patches with channel-independent processing across patches (Liu et al., 2025). Overall, these methods differ in how irregular observations are represented and at which stages and scales temporal and cross-channel information is integrated.

Graph neural networks for multi-channel time series. Graph-based methods differ in what their nodes represent and how their edges are constructed. MTGNN, Raindrop, GNeuralFlow, and WaveGNN learn channel-level dependency structures from data (Wu et al., 2020; Zhang et al., 2022; Mercatali et al., 2024; Hajisafi et al., 2025), whereas KEDGN incorporates medical text semantics and observation density into graph construction (Luo et al., 2024). Patch-based graphs organize interactions at different resolutions: t-PatchGNN connects channel-specific patch representations, while Hi-Patch builds hierarchical graphs over observations and aggregated patches (Zhang et al., 2024; Luo et al., 2025). GraFITi and TimeCHEAT instead encode observed values on edges connecting channel and time nodes, operating over entire windows and within temporal patches, respectively (Yalavarthi et al., 2024; Liu et al., 2025). ASTGI represents individual observations as nodes and constructs temporally constrained neighborhoods in a learned spatio-temporal space (Liu et al., 2026a). These constructions organize message passing through channel dependencies, channel-time relations, or observation-level neighborhoods, giving edges different meanings and determining which information can interact.

3 METHOD

3.1 PROBLEM FORMULATION

Let $\mathcal{S} = \{(\mathbf{s}_i,\ y_i)\}_{i=1}^{M}$ denote a dataset of M entity samples, where $\mathbf{s}_i$ is the i-th irregularly sampled multi-channel time series and y_i is its class label. Each entity has C channels, with at most T observations per channel. We store the observed values in a measurement matrix $\mathbf{X}_i \in \mathbb{R}^{T\times C}$ and indicate measurement availability using an observation mask $\mathbf{O}_i \in \{0,1\}^{T\times C}$: The timestamps $\tau_i \in \mathbb{R}^{T\times 1}$ record the actual sampling time of each measurement. We consider both sequence classification and forecasting. Classification predicts the sample label y_i from the complete observation history, whereas forecasting estimates future channel values from the available observations.

3.2 GRAPH CONSTRUCTION WITH REFERENCE ENTITIES

Rather than predefining pairwise channel relations, we construct a local bipartite graph at each irregular observation step t, with one real entity and K learnable reference entities connected to C channel nodes:

$$\mathcal{G}_t = \left(\{\mathbf{E}_t\} \cup \mathcal{U}, \mathcal{C}_t,\ \mathcal{E}_t^{real} \cup \mathcal{E}^{ref}\right). \tag{1}$$

Here, $\mathcal{U} = \{u_{t,k}\}_{k=1}^{K}$ is the set of shared reference entities, and $\mathcal{C}_t = \{c_{t,j}\}_{j=1}^{C}$ is the set of channel nodes for this sample at time t. Observation edges $\mathcal{E}_t^{real}$ are created only when measurements are available, whereas reference edges $\mathcal{E}^{ref}$ connect each channel to all K reference entities.

The observation mask $o_{j,t}$ determines connectivity for the real entity: an observation edge is created only when $o_{j,t} = 1$, indicating that channel j is measured at time t for the sample. Reference edges connect the channels of each sample to a fixed set of shared reference entities, providing a consistent population-level reference.

The graph propagates information through attributed edges. For each observed measurement, we separately encode its value, sampling time, and channel identity. The three encodings are first added and then multiplied by the observation mask to zero out missing entries, yielding the initial edge

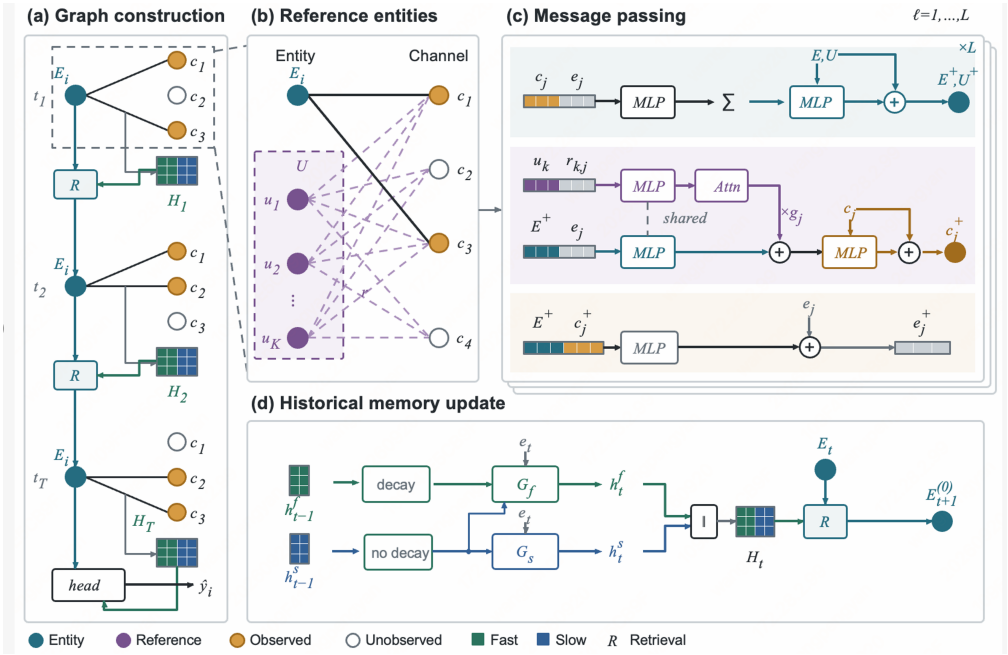

Figure 2: Overview of REA-DG. (a) At each step, observations determine which channels connect to the real entity. (b) Shared learnable reference entities provide context learned across training samples. (c) Observed channels update the real and reference entities; real-entity feedback combines with gated reference messages to update channels; the updated endpoints then refine observation edges. (d) Fast-slow memories retain channel history, with interval-dependent decay applied only to the fast memory.

state $\mathbf{e}_{t,j}^{in}$:

$$\mathbf{e}_{t,j}^{in} = o_{t,j} \ \left(\mathrm{Emb}_{value}\left(x_{j,t}\right) + \mathrm{Emb}_{time}\left(\tau_{j,t}\right) + \mathrm{Emb}_{channel}(j)\right) \in \mathbb{R}^D. \tag{2}$$

Here, Emb_{value} is a linear layer, $\mathrm{Emb}_{channel}(j)$ is a learnable embedding of channel j, and Emb_{time} is a Time2Vec-style periodic encoding (Kazemi et al., 2019) that concatenates a linear component with sinusoidal components:

$$\mathrm{Emb}_{time}(\tau) = \mathrm{cat}\left(w_0\tau + b_0,\ \sin(\mathbf{w}_1\tau + \mathbf{b}_1)\right). \tag{3}$$

The linear component preserves the ordering of absolute timestamps, while the sinusoidal components represent temporal positions and periodic patterns. Each edge therefore encodes the measurement value, its sampling time, and its channel identity. In the following, we denote the edge state during graph propagation by $\mathbf{e}_{t,j}$, initialized as $\mathbf{e}_{t,j}^{in}$.

Node embeddings are learnable vectors. Channel node j is initialized from its embedding $\mathbf{q}_j \in \mathbb{R}^D$, which identifies the channel, while the real entity is initialized from a shared vector $\mathbf{E}^{base} \in \mathbb{R}^D$. At the first observation step ($t = 1$), no historical observations are available, so the real entity node is initialized using the base embedding $\mathbf{E}^{base}$ shared across all samples.The channel node state $\mathbf{c}_{t,j}$ is initialized from the learnable channel embedding $\mathbf{q}_j$.

Messages from shared reference entities are generated from learnable entity embeddings $\mathbf{u}_k \in \mathbb{R}^D$ and edge embeddings $\mathbf{r}_{k,j} \in \mathbb{R}^D$, with one edge embedding per reference entity-channel pair.

Multilayer Entity-Channel Message Passing Within each observation step, we stack L graph layers to update the representations of the entities, channels, and observation edges. Each layer performs three operations in a fixed order. First, information from observed channels is aggregated to

update the entities. Next, the updated real entity provides sample-specific information to the channels, while the shared reference entities provide context learned across training samples. Finally, edge representations are updated using the new states of nodes.

Graph Structure and Message Passing Each graph layer updates the entities, channels, and observation edges in sequence, with reference entities providing shared context.

First, we project the concatenated channel and edge states and sum messages over observed channels. The resulting message updates the entities through a residual connection and layer normalization:

$$\begin{aligned}
\bar{\mathbf{m}}_t &= \sum_{j=1}^{C} O_{t,j}\,\mathrm{MLP}_{c\to E}\big([\mathbf{c}_{t,j}\|\mathbf{e}_{t,j}]\big),\\
\bar{\mathbf{m}}^U_{t,k} &= \sum_{j=1}^{C} O_{t,j}\,\mathrm{MLP}_{c\to E}\big([\mathbf{c}_{t,j}\|\mathbf{r}_{k,j}]\big),\\
\mathbf{E}^+_t &= \mathrm{LN}\big(\mathbf{E}_t + \mathrm{MLP}_E\big([\mathbf{E}_t\|\bar{\mathbf{m}}_t]\big)\big)\,,\\
\mathbf{u}^+_{t,k} &= \mathrm{LN}\big(\mathbf{u}_{t,k} + \mathrm{MLP}_E\big([\mathbf{u}_{t,k}\|\bar{\mathbf{m}}^U_{t,k}]\big)\big)\,.
\end{aligned} \tag{4}$$

Here, $O_{t,j}$ is the observation mask, $[\cdot\|\cdot]$ denotes concatenation, and LN denotes layer normalization. Superscript $+$ marks states updated within the current layer; sample and layer indices are omitted.

The updated real entity and the reference entities then send messages to channels through a shared projector . For reference entity $\mathbf{u}_{t,k}$ and its edge $\mathbf{r}_{k,j}$ to channel j, we compute reference messages, aggregate them with attention weights $\alpha_{k,j}$, and scale their contribution with a learnable channel-specific scalar gate g_j:

$$\begin{aligned}
\mathbf{m}^{\mathrm{real}}_{t,j} &= O_{t,j}\,\mathrm{MLP}_{E\to c}\big([\mathbf{E}^+_t\|\mathbf{e}_{t,j}]\big),\\
\mathbf{m}^{\mathrm{ref}}_{k,j} &= \mathrm{MLP}_{E\to c}\big([\mathbf{u}_{t,k}\|\mathbf{r}_{k,j}]\big),\\
\mathbf{m}^{\mathrm{in}}_{t,j} &= \mathbf{m}^{\mathrm{real}}_{t,j} + g_j \sum_{k=1}^{K} \alpha_{k,j}\mathbf{m}^{\mathrm{ref}}_{k,j}.
\end{aligned} \tag{5}$$

Here, $\alpha_{k,j} \geq 0$ and $\sum_{k=1}^{K} \alpha_{k,j} = 1$. The observation mask applies only to real-entity messages; unobserved channels can still receive reference context.

Channels incorporate the combined message $\mathbf{m}^{\mathrm{in}}_{t,j}$ through a residual update. Each observation edge is then updated using the new entity and channel states together with its current representation:

$$\begin{aligned}
\mathbf{c}^+_{t,j} &= \mathrm{LN}\big(\mathbf{c}_{t,j} + \mathrm{MLP}_c\big([\mathbf{c}_{t,j}\|\mathbf{m}^{\mathrm{in}}_{t,j}]\big)\big)\,,\\
\mathbf{e}^+_{t,j} &= \mathrm{LN}\big(\mathbf{e}_{t,j} + \mathrm{MLP}_e\big([\mathbf{E}^+_t\|\mathbf{c}^+_{t,j}\|\mathbf{e}_{t,j}]\big)\big)\,, \quad O_{t,j} = 1.
\end{aligned} \tag{6}$$

We stack L layers with the same structure and independent parameters, passing the updated entity, channel, and edge states to the next layer. The path $c_a \to \mathbf{E} \to c_b$ enables cross-channel interaction without first estimating a dense $C \times C$ channel graph. The final observation-edge representations are used for the subsequent channel-memory update.

Interval-Gated Memory Update Each channel maintains fast and slow memories, $\mathbf{h}^f_{t,j}, \mathbf{h}^s_{t,j} \in \mathbb{R}^{D/2}$. A shared learned projection maps the current edge representation $\mathbf{e}_{t,j} \in \mathbb{R}^D$ to $\bar{\mathbf{e}}_{t,j} = \mathrm{Proj}(\mathbf{e}_{t,j}) \in \mathbb{R}^{D/2}$ for both gated updates. Fast memory undergoes interval-dependent decay before this update, while slow memory does not.

For an observed channel, fast memory first decays as

$$\widetilde{\mathbf{h}}^f_{t-1,j} = \exp(-\lambda_{t,j}\Delta_{t,j})\,\mathbf{h}^f_{t-1,j}. \tag{7}$$

The positive decay rate is $\lambda_{t,j} = \mathrm{softplus}(\mathrm{MLP}_\lambda(\mathbf{d}_{t,j}))$, where $\mathbf{d}_{t,j} = [\mathbf{e}_{t,j}\|\mathbf{h}_{t-1,j}] \in \mathbb{R}^{2D}$ concatenates the original edge representation and the previous composite memory. Let t^- and t^+ denote

the nearest preceding and following observation times for channel j. We define the sampling interval as

$$\Delta_{t,j} = \begin{cases} \frac{1}{2}(t^+ - t^-), & \text{both neighbors exist,} \\ \frac{1}{2}(t - t^-), & \text{only } t^- \text{ exists,} \\ \frac{1}{2}(t^+ - t), & \text{only } t^+ \text{ exists,} \\ \frac{1}{2}T_{\max}, & \text{neither neighbor exists.} \end{cases} \tag{8}$$

Here, $T_{\max}$ is the channel's full-sequence maximum interval. For unobserved channels, we set $\Delta_{t,j} = 0$.

The two gates determine the contribution of current and historical information in each dimension:

$$\begin{aligned} \mathbf{g}^f_{t,j} &= \sigma\Big(\mathrm{MLP}_g\big([\bar{\mathbf{e}}_{t,j} \| \widetilde{\mathbf{h}}^f_{t-1,j} \| \mathbf{h}^s_{t-1,j}]\big)\Big), \\ \mathbf{g}^s_{t,j} &= \sigma\big(\mathrm{MLP}_\beta\big([\bar{\mathbf{e}}_{t,j} \| \mathbf{h}^s_{t-1,j}]\big)\big). \end{aligned} \tag{9}$$

Here, σ is the sigmoid function, $\|$ denotes concatenation, and both gates are $D/2$-dimensional vectors. The resulting memory updates are

$$\begin{aligned} \mathbf{h}^f_{t,j} &= \mathrm{LN}\Big((\mathbf{1} - \mathbf{g}^f_{t,j}) \odot \widetilde{\mathbf{h}}^f_{t-1,j} + \mathbf{g}^f_{t,j} \odot \bar{\mathbf{e}}_{t,j}\Big), \\ \mathbf{h}^s_{t,j} &= (\mathbf{1} - \mathbf{g}^s_{t,j}) \odot \mathbf{h}^s_{t-1,j} + \mathbf{g}^s_{t,j} \odot \bar{\mathbf{e}}_{t,j}. \end{aligned} \tag{10}$$

Here, $\odot$ denotes elementwise multiplication, and layer normalization LN is applied only to fast memory.

Observed channels receive the new states, while unobserved channels retain their previous states. The two memories are then concatenated for subsequent use:

$$\begin{aligned} &\mathbf{h}^k_{t,j} \leftarrow O_{t,j}\mathbf{h}^k_{t,j} + (1 - O_{t,j})\mathbf{h}^k_{t-1,j}, \qquad k \in \{f, s\}, \\ &\mathbf{h}_{t,j} = [\mathbf{h}^f_{t,j} \| \mathbf{h}^s_{t,j}] \in \mathbb{R}^D. \end{aligned} \tag{11}$$

The arrow $\leftarrow$ denotes an in-place state update.

Historical Memory Retrieval Although the graph is reconstructed from the current observations at each time step, the entity state carries information from its history. Before graph propagation at $t > 1$, the previous entity state queries accumulated channel memories to incorporate historical information into the current graph computation. Let $\mathcal{H}_{t-1} = \{j : \sum_{\tau<t} O_{\tau,j} > 0\}$ denote the set of channels observed at least once before step t. For nonempty $\mathcal{H}_{t-1}$, we compute

$$\begin{aligned} s_{t,j} &= \frac{(W_q \mathbf{E}^{(L)}_{t-1})^\top (W_k \mathbf{h}_{t-1,j})}{\sqrt{D}}, \qquad a_{t,j} = \frac{\exp(s_{t,j})}{\sum_{k \in \mathcal{H}_{t-1}} \exp(s_{t,k})}, \\ \mathbf{E}^{(0)}_t &= \mathrm{LN}\left(\mathbf{E}^{(L)}_{t-1} + W_c \sum_{j \in \mathcal{H}_{t-1}} a_{t,j} \mathbf{h}_{t-1,j}\right). \end{aligned} \tag{12}$$

Here, $j \in \mathcal{H}_{t-1}$, and W_q, W_k, W_c are learnable projections. The retrieved memory is added to the previous entity state $\mathbf{E}^{(L)}_{t-1}$, producing the initial state $\mathbf{E}^{(0)}_t$ for the current step's graph propagation. This allows the current graph computation to use both present observations and information accumulated across earlier steps. Channels never observed are excluded from retrieval; when no historical observations are available, the entity is initialized from the shared base embedding.

3.3 Task-Specific Prediction Heads

After the last observed step T, the backbone provides the entity state $\mathbf{E}_T$ and the channel memories $\{\mathbf{h}_{T,j}\}_{j=1}^C$. Two task heads read these outputs; both objectives share a regularizer that keeps the reference entities distinct.

Reference-message regularization. Attention over the K reference messages does not by itself keep them distinct. Let $\mathbf{m}_{k,j}^{(\ell)}$ be the message from reference entity k to channel j at layer ℓ, evaluated with dropout disabled so that random masking is not counted as diversity. We penalize pairs of messages to the same channel whose cosine similarity exceeds a margin ρ:

$$\mathcal{L}_{\cos} = \frac{1}{LCK(K-1)} \sum_{\ell=1}^{L} \sum_{j=1}^{C} \sum_{k \neq r} \left[\cos\left(\mathbf{m}_{k,j}^{(\ell)}, \mathbf{m}_{r,j}^{(\ell)}\right) - \rho \right]_+^2, \tag{13}$$

where $[x]_+ = \max(x, 0)$ and $\mathcal{L}_{\cos} = 0$ for $K = 1$. The margin leaves moderately similar messages unpenalized, and the term constrains message directions only, not the attention weights that read them. It enters both task objectives below with weight $\lambda_{\cos} \geq 0$.

Classification. Let $n_j = \sum_{t=1}^{T} o_{t,j}$ be the number of observations of channel j and $S_{T,j} = \mathbf{1}[n_j > 0]$. Memories of observed channels are reweighted by a masked softmax over their counts,

$$\omega_j = \frac{S_{T,j} \exp(n_j)}{\sum_{j'=1}^{C} S_{T,j'} \exp(n_{j'})}, \qquad \tilde{\mathbf{h}}_j = S_{T,j} \, (1 + \omega_j) \, \mathbf{h}_{T,j}, \tag{14}$$

so that every observed memory is retained, frequently observed channels contribute more, and unobserved channels are zero. The final entity state and the reweighted memories are concatenated and passed to an MLP classifier, trained with cross-entropy and the shared regularizer:

$$\hat{\mathbf{y}} = \mathrm{softmax}\big(f_{\mathrm{cls}}\big([\mathbf{E}_T \,\|\, \tilde{\mathbf{h}}_1 \,\|\, \cdots \,\|\, \tilde{\mathbf{h}}_C]\big)\big), \qquad \mathcal{L}_{\mathrm{cls}} = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \log \hat{y}_{i,y_i} + \lambda_{\cos} \mathcal{L}_{\cos}, \tag{15}$$

where $\mathcal{B}$ is a minibatch and y_i the label of sample i.

Forecasting. To predict future channel values, the forecasting head combines the final entity state, the reweighted channel memories, and the future query-time encoding. At each query time τ_q, the decoder jointly predicts all C channels:

$$\widehat{\mathbf{X}}(\tau_q) = f_{\mathrm{pred}}\Big(\Big[\mathbf{E}_T \| \widetilde{\mathbf{h}}_1 \| \cdots \| \widetilde{\mathbf{h}}_C \| \, \mathrm{Emb}_{\mathrm{time}}(\tau_q)\Big]\Big) \in \mathbb{R}^C. \tag{16}$$

Here, $\widehat{X}_j(\tau_q)$ denotes the predicted value of channel j at future query time τ_q.

Only observed future targets contribute to the forecasting error. For sample i, let $\Omega_{\mathrm{fut},i}$ denote the set of channel-query pairs (j, q) with available ground truth, and let $\mathcal{B}_{\mathrm{fut}} = \{i \in \mathcal{B} : |\Omega_{\mathrm{fut},i}| > 0\}$ denote the samples in the minibatch with at least one observed future target. We average squared errors over the valid targets within each sample and then over these samples:

$$\mathcal{L}_{\mathrm{pred}} = \frac{1}{|\mathcal{B}_{\mathrm{fut}}|} \sum_{i \in \mathcal{B}_{\mathrm{fut}}} \frac{1}{|\Omega_{\mathrm{fut},i}|} \sum_{(j,q) \in \Omega_{\mathrm{fut},i}} \left(\widehat{X}_{i,j}(\tau_{i,q}) - X_{i,j}(\tau_{i,q}) \right)^2, \tag{17}$$

where $\tau_{i,q}$ is the q-th future query time of sample i. When reference-message regularization is used, the total forecasting objective is

$$\mathcal{L}_{\mathrm{total}}^{\mathrm{pred}} = \mathcal{L}_{\mathrm{pred}} + \lambda_{\cos} \mathcal{L}_{\cos}. \tag{18}$$

4 Experiments

4.1 Experimental Setup

For clinical classification, we evaluate on P19, P12, PhysioNet, and MIMIC-III using patient-disjoint training, validation, and test sets. Following the split settings of KEDGN (Luo et al., 2024). FNSPID uses six numerical channels with 31-day input and prediction windows, while CESNET uses ten numerical channels with seven-day input and prediction windows.

For REA-DG, classification checkpoints are selected by the highest validation AUPRC. We report mean $\pm$ standard deviation over five independent runs on each dataset. Classification is evaluated

Table 1: Comparison of the classification performance P19, PhysioNet, MIMIC-III and P12.

	P19		PhysioNet		MIMIC-III		P12	
Method	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
GRU-D	88.7 ± 1.2	57.6 ± 2.3	79.1 ± 6.9	42.7 ± 7.2	82.2 ± 1.8	43.3 ± 2.1	79.6 ± 0.6	41.7 ± 1.8
ODE-RNN	87.1 ± 1.0	52.6 ± 3.2	75.5 ± 2.8	33.7 ± 4.1	81.0 ± 0.6	42.3 ± 0.7	78.8 ± 0.6	37.4 ± 2.6
IP-Net	90.2 ± 0.2	58.6 ± 0.8	86.8 ± 0.6	55.8 ± 1.4	84.1 ± 0.1	47.1 ± 0.9	83.7 ± 0.3	46.3 ± 1.3
SeFT	84.0 ± 0.3	49.3 ± 0.5	75.5 ± 0.2	29.4 ± 0.9	67.9 ± 0.2	23.2 ± 0.4	78.1 ± 0.5	35.9 ± 0.8
MTGNN	88.5 ± 1.0	55.8 ± 1.5	77.1 ± 4.4	35.4 ± 7.3	78.5 ± 2.3	35.2 ± 3.1	82.1 ± 1.5	41.8 ± 2.1
mTAND	82.9 ± 0.9	32.2 ± 1.5	86.9 ± 1.3	52.5 ± 1.3	83.8 ± 0.3	46.6 ± 0.5	85.3 ± 0.3	49.3 ± 1.0
DGM2-O	91.6 ± 0.5	60.0 ± 1.3	85.8 ± 0.7	50.4 ± 3.2	81.0 ± 0.9	37.6 ± 1.1	85.8 ± 0.1	48.3 ± 0.7
Raindrop	89.4 ± 0.6	61.2 ± 1.1	82.7 ± 1.4	41.2 ± 3.6	79.8 ± 1.3	35.2 ± 1.1	82.2 ± 1.1	43.3 ± 2.1
STraTS	91.2 ± 0.3	58.4 ± 1.4	84.9 ± 1.5	47.3 ± 5.3	84.4 ± 0.4	46.4 ± 0.8	86.7 ± 0.7	52.1 ± 1.5
DuETT	88.2 ± 0.5	56.0 ± 3.9	81.3 ± 1.4	44.9 ± 1.4	78.8 ± 0.8	34.3 ± 1.0	83.4 ± 1.2	45.4 ± 1.5
ViTST	91.7 ± 0.1	57.5 ± 0.7	81.3 ± 1.9	37.4 ± 2.9	81.8 ± 0.3	39.6 ± 1.3	86.3 ± 0.1	50.8 ± 1.5
Warpformer	91.8 ± 0.4	60.6 ± 2.6	83.3 ± 0.7	43.5 ± 2.3	84.6 ± 0.5	47.4 ± 0.9	85.4 ± 0.5	50.4 ± 1.5
Hi-Patch	92.1 ± 0.4	61.1 ± 2.1	86.8 ± 0.9	57.3 ± 1.9	84.8 ± 0.2	47.2 ± 1.0	86.9 ± 0.7	53.3 ± 0.9
TimeCHEAT	89.5 ± 1.9	56.1 ± 4.6	-	-	-	-	84.5 ± 0.7	48.2 ± 1.9
KEDGN	92.3 ± 1.0	62.5 ± 0.7	87.9 ± 1.4	56.0 ± 3.2	84.8 ± 0.3	48.4 ± 1.5	87.1 ± 0.8	54.1 ± 2.6
REA-DG (Ours)	$\mathbf{94.5 \pm 0.3}$	$\mathbf{71.3 \pm 0.5}$	$\mathbf{88.6 \pm 0.2}$	$\mathbf{58.8 \pm 1.1}$	$\mathbf{85.7 \pm 0.6}$	$\mathbf{52.2 \pm 1.9}$	$\mathbf{87.9 \pm 0.3}$	$\mathbf{55.4 \pm 0.8}$

using AUROC and AUPRC, while forecasting uses normalized-space MSE and MAE computed on observed future targets. Detailed dataset construction, preprocessing, training and evaluation protocols, and hyperparameter settings are provided in Appendix A.

We report AUROC/AUPRC for classification and normalized-space MSE/MAE for forecasting, as mean $\pm$ standard deviation. Model selection uses validation AUPRC and MSE, respectively. Dataset construction, training settings, and reporting details are provided in Appendix A and B.

4.2 Experimental Results

As shown in Table 1, REA-DG achieves the highest mean AUROC and AUPRC among the compared methods on all four clinical datasets. Its AUPRC reaches 71.3% on P19 and 52.2% on MIMIC-III, exceeding the strongest respective baselines by 8.8 and 3.8 percentage points, with additional gains of 1.5 and 1.3 percentage points on PhysioNet and P12. Under the reported forecasting protocols, Table 2 shows that REA-DG achieves the lowest mean MSE and MAE on both forecasting datasets. On FNSPID, its MSE and MAE are 0.1095 and 0.1952, corresponding to reductions of 7.8% and 7.0% relative to Hi-Patch, respectively. On CESNET, the corresponding errors are 0.8677 and 0.7066, representing reductions of 3.0% and 1.4% relative to TimeCHEAT. Table 4 further shows that REA-DG achieves the lowest mean cumulative MSE over the first 10, 50, and 100 forecast hours on CESNET. Together, these results across domains, tasks, and forecast intervals support the effectiveness of jointly modeling current observations, sample-specific history, and shared context. By organizing interactions among observed channels through a sample entity and incorporating channel memories and reference entities, REA-DG integrates limited evidence scattered across channels and time into representations that support both classification and forecasting, providing an effective approach to information integration under sparse and asynchronous observations.

4.3 Ablation Studies

As shown in Table 3, the full model achieves the highest mean AUROC (94.51%) and AUPRC (71.37%) on P19. Removing reference entities lowers these metrics by 1.25 and 1.34 percentage points, respectively, supporting the value of shared context in supplementing limited sample-specific information. Among the tested reference counts, $K = 4$ performs best; fewer references yield lower performance, while $K = 8$ provides no further gains, indicating that a small set suffices to provide effective shared context. Removing entity–channel graph message passing reduces AUROC and AUPRC by 4.19 and 11.11 percentage points, highlighting the importance of observation-driven cross-channel interactions for integrating sparse evidence. Removing slow or fast memory lowers AUPRC by 0.68 and 3.12 percentage points, respectively, supporting their complementary roles

Table 2: Comparison of the forecasting results on FNSPID and CESNET.

	FNSPID		CESNET	
Method	MSE	MAE	MSE	MAE
GNeuralFlow	0.6950 ± 0.1249	0.5757 ± 0.0545	1.0045 ± 0.0027	0.7844 ± 0.0016
CRU	0.1947 ± 0.0125	0.3098 ± 0.0144	1.0046 ± 0.0021	0.7870 ± 0.0014
Neural Flows	0.1815 ± 0.0033	0.2923 ± 0.0055	1.0329 ± 0.0114	0.7948 ± 0.0054
TimeCHEAT	0.1734 ± 0.0134	0.2836 ± 0.0192	0.8944 ± 0.0315	0.7166 ± 0.0144
mTAN	0.1463 ± 0.0079	0.2463 ± 0.0100	1.0016 ± 0.0001	0.7862 ± 0.0004
GraFITi	0.1408 ± 0.0149	0.2461 ± 0.0237	0.9934 ± 0.0016	0.7806 ± 0.0008
Latent ODE	0.1300 ± 0.0045	0.2293 ± 0.0064	0.9626 ± 0.0012	0.7628 ± 0.0015
ASTGI	0.1293 ± 0.0131	0.2288 ± 0.0223	0.9985 ± 0.0011	0.7831 ± 0.0010
t-PatchGNN	0.1225 ± 0.0035	0.2154 ± 0.0055	0.9936 ± 0.0025	0.7807 ± 0.0016
Hi-Patch	0.1187 ± 0.0027	0.2098 ± 0.0056	0.9721 ± 0.0049	0.7688 ± 0.0029
REA-DG (Ours)	$\mathbf{0.1095 \pm 0.0013}$	$\mathbf{0.1952 \pm 0.0030}$	$\mathbf{0.8677 \pm 0.0006}$	$\mathbf{0.7066 \pm 0.0003}$

in modeling sample history. Together, these results support jointly modeling current observations, sample history, and shared context to integrate sparse and asynchronous observations.

Table 3: Ablation results of REA-DG on P19.

Method	Test AUROC	Test AUPRC
$K = 0$	93.26 ± 0.28	70.03 ± 0.26
$K = 1$	93.35 ± 0.31	70.44 ± 0.88
$K = 2$	93.98 ± 0.13	70.53 ± 0.34
$K = 8$	93.98 ± 0.24	70.43 ± 0.49
w/o Graph	90.32 ± 0.16	60.26 ± 0.41
w/o slow	93.49 ± 0.28	70.69 ± 1.27
w/o fast	93.63 ± 0.28	68.25 ± 0.77
Full, $K = 4$	$\mathbf{94.51 \pm 0.31}$	$\mathbf{71.37 \pm 0.51}$

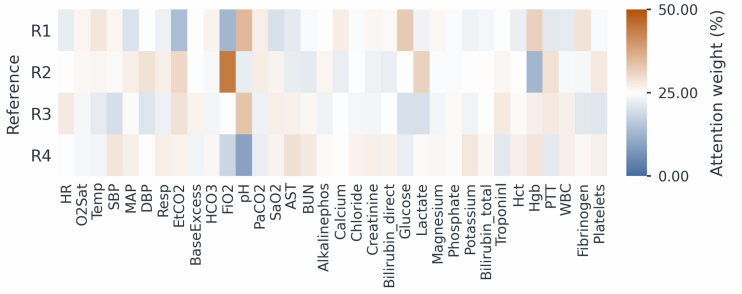

Figure 3: Reference-to-channel attention on P19 ($K = 4$).

5 Conclusion

We presented REA-DG for irregularly sampled multi-variate time series. REA-DG connects each sample entity to its currently observed channels, integrating cross-channel interaction, shared reference context, and sample-specific history maintained by fast and slow channel memories. Experiments on classification and forecasting tasks support its effectiveness, while ablation studies show the benefits of graph message passing, reference entities, and interval-dependent memory decay.

AI Use Statement

The authors used LLMs to improve the readability of the manuscript, including checking grammar, punctuation, and syntax; reducing redundancy; clarifying sentence structure; and improving transitions and overall flow.

REFERENCES

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

Ching Chang, Jeehyun Hwang, Yidan Shi, Haixin Wang, Wei Wang, Wen-Chih Peng, and Tien-Fu Chen. Time-IMM: A Dataset and Benchmark for Irregular Multimodal Multivariate Time Series. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025.

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), April 2018. ISSN 2045-2322.

Robert F. Engle and Jeffrey R. Russell. Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data. *Econometrica*, 66(5):1127–1162, September 1998.

Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, 101(23), June 2000. ISSN 1524-4539.

Arash Hajisafi, Maria Despoina Siampou, Bita Azarijoo, Zhen Xiong, and Cyrus Shahabi. WaveGNN: Integrating Graph Neural Networks and Transformers for Decay-Aware Classification of Irregular Clinical Time-Series. In *2025 IEEE International Conference on Big Data (BigData)*, pp. 1934–1943. IEEE, December 2025.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set Functions for Time Series. In *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.

Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

Josef Koumar and Tomas Čejka. Unevenly Spaced Time Series from Network Traffic. In *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–4. IEEE, June 2023.

Alex Labach, Aslesha Pokhrel, Xiao Shi Huang, Saba Zuberi, Seung Eun Yi, Maksims Volkovs, Tomi Poutanen, and Rahul G. Krishnan. DuETT: Dual Event Time Transformer for Electronic Health Records. In *Proceedings of the 8th Machine Learning for Healthcare Conference*, volume 219 of *Proceedings of Machine Learning Research*, pp. 403–422. PMLR, 11–12 Aug 2023.

Zekun Li, Shiyang Li, and Xifeng Yan. Time Series as Images: Vision Transformer for Irregularly Sampled Time Series. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49187–49204. Curran Associates, Inc., 2023.

Jiexi Liu, Meng Cao, and Songcan Chen. TimeCHEAT: A Channel Harmony Strategy for Irregularly Sampled Multivariate Time Series Analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):18861–18869, April 2025. ISSN 2159-5399.

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 *The Fourteenth 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(28):23873–23881, March 2026b. ISSN 2159-5399.

Yicheng Luo, Zhen Liu, Linghao Wang, Junhao Zheng, Binquan Wu, and Qianli Ma. Knowledge-Empowered Dynamic Graph Network for Irregularly Sampled Medical Time Series. In *Advances in Neural Information Processing Systems*, volume 37, pp. 67172–67199. Curran Associates, Inc., 2024.

Yicheng Luo, Bowen Zhang, Zhen Liu, and Qianli Ma. Hi-Patch: Hierarchical Patch GNN for Irregular Multivariate Time Series. In *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.

Giangiacomo Mercatali, Andre Freitas, and Jie Chen. Graph neural flows for unveiling systemic interactions among irregularly sampled time series. In *Advances in Neural Information Processing Systems*, volume 37, pp. 57183–57206. Curran Associates, Inc., 2024.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*, 2023.

Matthew A. Reyna, Christopher S. Josef, Russell Jeter, Supreeth P. Shashikumar, M. Brandon Westover, Shamim Nemati, Gari D. Clifford, and Ashish Sharma. Early Prediction of Sepsis From Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019. *Critical Care Medicine*, 48(2):210–217, February 2020. ISSN 0090-3493.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling Irregular Time Series with Continuous Recurrent Units. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19388–19405. PMLR, 17–23 Jul 2022.

Satya Narayan Shukla and Benjamin Marlin. Interpolation-Prediction Networks for Irregularly Sampled Time Series. In *International Conference on Learning Representations*, 2019.

Satya Narayan Shukla and Benjamin Marlin. Multi-Time Attention Networks for Irregularly Sampled Time Series. In *International Conference on Learning Representations*, 2021.

Sindhu Tipirneni and Chandan K. Reddy. Self-Supervised Transformer for Sparse and Irregularly Sampled Multivariate Clinical Time-Series. *ACM Transactions on Knowledge Discovery from Data*, 16(6):1–17, July 2022. ISSN 1556-472X.

Yinjun Wu, Jingchao Ni, Wei Cheng, Bo Zong, Dongjin Song, Zhengzhang Chen, Yanchi Liu, Xuchao Zhang, Haifeng Chen, and Susan B Davidson. Dynamic Gaussian mixture based deep generative model for robust forecasting on sparse multivariate time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):651–659, 2021.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 753–763. ACM, August 2020.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(15):16255–16263, March 2024. ISSN 2159-5399.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, June 2023. ISSN 2159-5399.

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. ACM, August 2023.

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular multivariate time series forecasting: A transformable patching graph neural networks approach. In *ICML*, pp. 60179–60196, 2024.

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.

APPENDIX

A ADDITIONAL EXPERIMENTAL DETAILS

A.1 CLINICAL CLASSIFICATION

We conduct experiments on four widely used irregular clinical time series datasets: P19, which contains 38,803 entities, each with 34 channels (variables or sensors); P12; PhysioNet; and MIMIC-III. All datasets are partitioned into mutually exclusive subsets at the entity (patient) level, so that no entity appears in more than one of the training, validation, and test sets. P19 and P12 use the fixed split-1, PhysioNet uses split-5 to align with the official KEDGN experiments, and MIMIC-III uses predefined training, validation, and test splits. All normalization statistics are estimated exclusively from the training set and then fixed for application to the validation and test sets to avoid data leakage. The first three datasets use approximate split ratios of 80%/10%/10%, whereas MIMIC-III uses approximately 69.47%/15.24%/15.29%. The positive sample counts in the table also indicate substantial class imbalance in all four tasks. Accuracy alone is therefore insufficient to characterize the models' ability to identify the minority positive class.

A.2 CLASSIFICATION TRAINING

Each clinical classification experiment uses a single GPU. Experiments on P19, PhysioNet, P12, and MIMIC-III are conducted on NVIDIA A100 80GB PCIe GPUs.

For these four clinical classification datasets, we use the Adam optimizer and cross-entropy loss and train for a maximum of 30 epochs. We evaluate the model on the validation set after each epoch and retain the checkpoint with the highest validation AUPRC. P12 additionally uses early stopping with a patience of 5 epochs. For the other three datasets, training proceeds to the maximum number of epochs, and the checkpoint with the best validation performance is evaluated once on the test set. We use a class-balanced batch sampler for training and do not shuffle the validation or test sets. The batch size is 256 for training, validation, and testing.

We use $K = 4$ reference entities for all four clinical classification datasets.

Each clinical classification task is run independently five times using random seeds 0, 1, 2, 3, and 4.

A.3 FORECASTING DATA AND PREPROCESSING

FNSPID uses six numerical channels, with 31 days of history, 31 days of prediction, and a 31-day window stride. CESNET uses ten channels, with seven days of history, seven days of prediction, and a seven-day stride. Windows are ordered chronologically within each entity and partitioned using the first $\lfloor 0.6n \rfloor$ windows for training, the next $\lfloor 0.8n \rfloor - \lfloor 0.6n \rfloor$ for validation, and the remaining windows for testing, where n is the number of retained windows for that entity. These are within-entity window splits, not patient-disjoint classification splits or a common calendar-date split across all entities. The CESNET subset contains 11 entities and 23/11/12 training/validation/test windows.

Both forecasting datasets retain explicit observation masks and timestamps. Padding does not count as an observation, and valid zero or negative values are retained. Future target values are used only for training losses and evaluation, not as model inputs. No news text or text embeddings are supplied to the forecasting model, although the upstream window-selection rule based on text availability is retained.

A.4 FORECASTING TRAINING AND REPORTING

For FNSPID, REA-DG is trained with Adam, a learning rate of 2.5×10^{-3}, weight decay of 10^{-4}, and batch size 256, for at most 100 epochs. Early stopping monitors validation MSE with patience 10. For CESNET, the time-basis query-head configuration uses Adam with a learning rate of 10^{-3}, weight decay of 10^{-3}, batch size 16, at most 300 epochs, and validation-MSE early stopping with patience 30. This query head uses hidden width 64 and fixed multiscale time features with frequencies $1, 2, 4, 8, 16, 32$.

For each forecasting run of REA-DG, the three checkpoints with the lowest validation MSE are averaged in parameter space, and the resulting model is evaluated once on the test set. The FNSPID result aggregates seeds $0, 1, 2, 3, 4$; the CESNET result aggregates seeds$0, 1, 2, 3, 4$. These counts describe REA-DG and do not imply that every baseline uses the same seed count. The CESNET row in tab. 2 uses the time-basis query head; the original-head control is not included in that main-results table.

Forecasting MSE and MAE are computed only on observed future targets. Errors and valid-target counts are pooled across test windows separately for each channel, and the resulting channel-wise errors are averaged across channels with valid targets.

B Evaluation Metrics and Reporting

AUROC measures the probability that a randomly selected positive sample receives a higher prediction score than a randomly selected negative sample. Let N+ and N− denote the numbers of positive and negative test samples, respectively, and let R denote the ranks of positive prediction scores among all samples. AUROC is computed from the sum of the positive samples' ranks, with tied scores assigned average ranks. This metric assesses overall ranking ability and does not depend on a fixed classification threshold.

In our implementation, AUPRC is computed as average precision. All samples are sorted in descending order of their predicted positive-class scores. Whenever a true positive sample is encountered, precision is recorded over all samples up to that position. These precision values are then averaged over all positive-sample positions. AUPRC therefore places greater emphasis on positive samples ranked near the top and is more informative than accuracy for clinical tasks with a low prevalence of positive cases.

For a metric with values $m_1, \ldots, m_R$ from R independent runs, we report

$$\bar{m} = \frac{1}{R} \sum_{r=1}^{R} m_r, \tag{19}$$

$$s = \sqrt{\frac{1}{R} \sum_{r=1}^{R} (m_r - \bar{m})^2}. \tag{20}$$

Here, $R = 5$ for REA-DG classification and FNSPID results, and $R = 5$ for the CESNET result. We use the population standard deviation (denominator R). Only AUROC and AUPRC are multiplied by 100 and reported as percentages; forecasting MSE and MAE remain in the normalized target space. A best single-seed score must not be paired with the standard deviation across multiple seeds.

C Additional Forecasting Analysis

C.1 Errors over Cumulative Forecast Intervals on CESNET

We further evaluate REA-DG across different cumulative portions of the CESNET forecast window. All models use 168 hours of history to predict the following 168 hours. Using these saved predictions, we compute errors over observed targets within the first $h \in \{10, 50, 100\}$ forecast hours. Each interval $[0, h]$ is measured from the end of the history window.

The evaluation uses 12 test windows from the 11-entity CESNET subset with ten numerical channels. The three intervals contain 980, 5,100, and 10,240 valid scalar targets, respectively. For each seed, errors are computed in normalized space, pooled over observed targets within each channel, and then averaged across the ten channels. We report means and population standard deviations across seeds. Neural Flows, CRU, and Latent ODE use seeds 0-5, while the remaining methods use seeds 0-5. REA-DG uses the multiscale time-basis prediction head employed in the main CESNET comparison.

Table 4 shows that REA-DG achieves the lowest mean cumulative MSE over the first 10, 50, and 100 forecast hours. These results extend the main comparison by demonstrating competitive performance across multiple evaluation intervals and further support the effectiveness of jointly modeling

Table 4: Cumulative MSE (↓) on CESNET over the first 10, 50, and 100 forecast hours, computed from saved 168-hour forecasts. Values are means ± population standard deviations; n denotes the number of seeds. The lowest mean in each column is in bold.

Method	n	$[0, 10]$ hours	$[0, 50]$ hours	$[0, 100]$ hours
GNeuralFlow	5	1.2200 ± 0.0095	0.9775 ± 0.0031	0.9952 ± 0.0023
CRU	5	1.2208 ± 0.0080	0.9773 ± 0.0036	0.9938 ± 0.0025
Neural Flows	5	1.2483 ± 0.0117	1.0120 ± 0.0163	1.0241 ± 0.0141
TimeCHEAT	5	1.2127 ± 0.0136	0.8881 ± 0.0272	0.9079 ± 0.0274
mTAN	5	1.2064 ± 0.0003	0.9708 ± 0.0002	0.9875 ± 0.0003
GraFITi	5	1.2061 ± 0.0004	0.9677 ± 0.0008	0.9863 ± 0.0005
Latent ODE	5	1.2835 ± 0.0115	0.9554 ± 0.0022	0.9767 ± 0.0015
ASTGI	5	1.2059 ± 0.0008	0.9719 ± 0.0013	0.9892 ± 0.0016
t-PatchGNN	5	1.2068 ± 0.0010	0.9668 ± 0.0013	0.9855 ± 0.0007
Hi-Patch	5	1.2269 ± 0.0063	0.9510 ± 0.0030	0.9755 ± 0.0019
REA-DG (Ours)	5	$\mathbf{1.2041 \pm 0.0004}$	$\mathbf{0.8614 \pm 0.0003}$	$\mathbf{0.8848 \pm 0.0003}$

current observations, sample-specific history, and shared context. By organizing observation-driven channel interactions through a sample entity and incorporating channel memories and reference entities, REA-DG integrates limited evidence scattered across channels and time into effective predictive representations, supporting its use for forecasting under sparse and asynchronous observations.