# GRAPH-GUIDED NETWORK FOR IRREGULARLY SAMPLED MULTIVARIATE TIME SERIES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In many domains, including healthcare, biology, and climate science, time series are irregularly sampled with variable time between successive observations and different subsets of variables (sensors) are observed at different time points, even after alignment to start events. These data create multiple challenges for prevailing models that assume fully observed and fixed-length feature representations. To address these challenges, it is essential to understand the relationships between sensors and how they evolve over time. Here, we introduce RAINDROP, a graph-guided network for learning representations of irregularly sampled multivariate time series. RAINDROP represents every sample as a graph, where nodes indicate sensors and edges represent dependencies between them. RAINDROP models dependencies between sensors using neural message passing and temporal self-attention. It considers both inter-sensor relationships shared across samples and those unique to each sample that can vary with time, and it adaptively estimates misaligned observations based on nearby observations. We use RAINDROP to classify time series and interpret temporal dynamics of three healthcare and human activity datasets. RAINDROP outperforms state-of-the-art methods by up to 11.4% (absolute points in F1 score), including methods that deal with irregular sampling using fixed discretization and set functions, and even in challenging leave-sensor-out settings and setups that require generalizing to new patient groups.

## 1 INTRODUCTION

Multivariate time series are prevalent in a variety of domains including healthcare, space science, cybersecurity, biology, and finance (Ravuri et al., 2021; Sousa et al., 2020; Sezer et al., 2020; Fawaz et al., 2019; Abanda et al., 2019; Tang et al., 2018). Practical issues often exist in collecting sensor measurements that lead to various types of irregularities caused by missing observations, such as cost saving, sensor failures, external forces in physical scenarios, medical interventions, to name a few (Choi et al., 2020). While temporal machine learning models usually assume fully observable and fixed-size input data, irregularly sampled time series raise considerable challenges. For example, the observations of multiple sensors are not well-aligned; the time intervals among adjacent observations are different across sensors; and different samples have different numbers of observations for different subsets of sensors recorded at different time points.

Prior methods for dealing with irregularly sampled time series involve filling in missing values, using interpolation, kernel methods, and probabilistic approaches (Schafer & Graham, 2002). The missingness of observations carry informative power (Little & Rubin, 2014) and thus imputation of missing data is not always beneficial (Agniel et al., 2018). While modern techniques involve recurrent neural network architectures (*e.g.*, RNN, LSTM, GRU) (Hochreiter & Schmidhuber, 1997; Cho et al., 2014) and transformers (Vaswani et al., 2017), they are restricted to regular sampling or assume aligned measurements across modalities. For misaligned measurements, existing methods tend to rely on a two-stage process of imputation to get a regularly-sampled version of a dataset, and then performing a downstream task such as classification. This decoupled approach might not fully exploit informative missingness patterns that could be essential for the downstream task thus achieving suboptimal performance (Wells et al., 2013; Li & Marlin, 2016). Thus, several approaches have been proposed that circumvent imputation and directly model irregularly sampled time series data (Che et al., 2018; Horn et al., 2020; Shukla & Marlin, 2021). However, none of them considered

relational structures in representation learning to address the characteristics of irregularly sampled multivariate time series.

**Present work.** To address the characteristics of irregularly sampled time series data, we propose to model temporal dynamics of sensor dependencies and how those relationships evolve over time. We develop RAINDROP[1], a graph-guided network that leverages relational structure to embed and classify irregularly sampled multivariate time series. RAINDROP takes *samples* as input, each sample containing multiple *sensors* and each sensor consisting of irregularly recorded *observations* (*e.g.*, in clinical data, an individual patient's state of health, recorded at irregular time intervals with different subsets of sensors observed at different times). RAINDROP model is inspired by the idea of raindrops falling into a pool at sequential but non-uniform time intervals and thereby creating ripple effects that propagate across the pool. Mathematically, in RAINDROP, observations (*i.e.*, raindrops) hit a sensor graph (*i.e.*, pool) asynchronously and at irregular time intervals, and each observation is processed by passing messages to neighboring sensors (*i.e.*, causes a ripple effect in the pool) taking into account the learned sensor dependencies (Figure 1). As such, RAINDROP can handle misaligned observations, varying time gaps, arbitrary numbers of observations, and produce embeddings via a novel hierarchical attention.

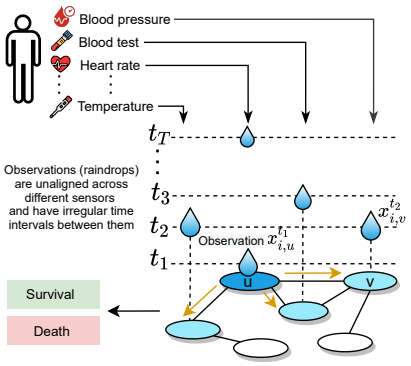



**Figure 1:** The RAINDROP approach. For sample $\mathcal{S}_i$, sensor $u$ is recorded at time $t_1$ as value $x_{i,u}^{t_1}$, triggering the propagation and non-linear transformation of neural messages along edges of $\mathcal{S}_i$'s sensor dependency graph.

RAINDROP is the first to explicitly model sensor dependencies in learning representations of irregularly sampled time series. We represent dependencies with a separate sensor graph for every sample, wherein nodes indicate sensors and edges denote relationships between them. In addition to capturing distinct sensor dependencies within each sample, RAINDROP i) takes advantage of similarities between different samples via sharing of parameters in the calculation of attention weights, and ii) considers relations between successive observations via temporal attention. RAINDROP adaptively estimates missing observations based on recorded information and the learned graph structure. We compare RAINDROP to five state-of-the-art methods across three datasets and four experimental settings, including a setup where a subset of sensors in the test set have malfunctioned (*i.e.*, have no readouts at all). Experiments show that RAINDROP outperforms baselines on all datasets with an average AUROC improvement of 3.5% in absolute points on classification tasks. Further, RAINDROP achieves a considerable margin (9.3% absolute points in accuracy on activity recognition) when a subset of sensors malfunction.

## 2 RELATED WORK

**Learning with irregularly sampled multivariate time series.** Irregular time series indicate that the time intervals between adjacent observations are varying (Zerveas et al., 2021; Tipirneni & Reddy, 2021; Chen et al., 2020). In the multivariate case, irregularity means that observations are misaligned across different sensors. Further, because of a multitude of sampling frequencies and varying time intervals, the number of observations can vary considerably across samples (Fang & Wang, 2020; Kidger et al., 2020). Predominant downstream tasks for time series are classification (*i.e.*, predicting a label for a given sample, *e.g.*, Tan et al. (2020); Ma et al. (2020)) and forecasting (*i.e.*, anticipating future observations based on historical observations, *e.g.*, Wu et al. (2020a)). The above mentioned data characteristics create considerable challenges for machine learning models that expect well-aligned and fixed-size input (Shukla & Marlin, 2020). An intuitive way to deal with irregular time series is to impute missing values and process them as a regular time series dataset (Mikalsen et al., 2021; Li & Marlin, 2020; Shan & Oliva, 2021). However, imputation methods can distort underlying distributions and introduce unwanted bias in the dataset. To this end, recent methods directly learn from irregularly sampled time series. For example, Che et al. (2018) propose a decay mechanism based on gated recurrent units (*GRU-D*), incorporating binary masking

[1]Code and datasets are available at https://anonymous.4open.science/r/Raindrop.

indicators and time intervals to capture long-term temporal dependencies. *SeFT* (Horn et al., 2020) re-formulates irregularly sampled time series into a set of observations and learn representations through set functions that are insensitive to alignment. Chen et al. (2018) model continuous-time hidden dynamics by latent ordinary differential equations (*Latent-ODE*), combined with an RNN for temporal representations. *mTAND* (Shukla & Marlin, 2021) leverages multi-time attention mechanism to learn temporal similarity from non-uniformly collected measurements and produces continuous-time embeddings. In contrast with these methods, RAINDROP leverages graphs to address the characteristics of irregular time series and improve the quality of learned representations.

**Learning with graphs and neural message passing.** There has been a surge of interest in applying neural networks to graphs, leading to the development of graph embeddings (Zhou et al., 2020; Li et al., 2021), graph neural networks (Wu et al., 2020b), and message passing neural networks (Gilmer et al., 2017). Our approach is mainly related to methods that perform message passing along edges to update node representations through neural transformations (Riba et al., 2018; Nikolentzos et al., 2020; Galkin et al., 2020; Fey et al., 2020; Lin et al., 2018; Zhang et al., 2020). However, in contrast to message passing used to make predictions on graphs, we are interested in leveraging it to meet the challenges of irregularly sampled time series. In particular, we consider message passing where nodes are sensors, describing a particular sample (*e.g.*, patient, Figure 1), and we design a message-passing network with learnable adjacency matrices. The key difference with the predominant use of message passing is that RAINDROP uses it to estimate edges (dependencies) between sensors rather than applying it on a fixed, apriori-given graph. To the best of our knowledge, no prior work utilized sensor dependencies for learning representations of irregularly sampled multivariate time series. Finally, while prior work used message passing for regular time series (Wang et al., 2020; Wu et al., 2020c; Kalinicheva et al., 2020), no modeling of irregularly sampled time series has been attempted.

## 3 RAINDROP

Let $\mathcal{D} = \{(\mathcal{S}_i, y_i) \mid i = 1, \ldots, N\}$ denote an irregular time series dataset with $N$ labeled samples (Figure 2). Every sample $\mathcal{S}_i$ is an irregular multivariate time series with a corresponding label $y_i \in \{1, \ldots, C\}$, indicating which of the $C$ classes $\mathcal{S}_i$ is associated with. Each sample contains multiple non-uniformly measured sensors that are denoted as $u$, $v$, etc. The set of sensors does not need to be the same across samples (see Sec. 4.1). Each sensor is given by a sequence of observations ordered by time. For sensor $u$ in sample $\mathcal{S}_i$, we denote a single observation as a tuple $(t, x_{i,u}^t)$, meaning that sensor $u$ was recorded with value $x_{i,u}^t \in \mathbb{R}$ at timestamp $t \in \mathbb{R}^+$. We omit sample index $i$ and sensor index $u$ in timestamp $t$. Sensor observations are irregularly recorded, meaning that time intervals between successive observations can vary across sensors. For sensor $u$ in sample $\mathcal{S}_i$, we use $\mathcal{T}_{i,u}$ to denote the set of timestamps that $u$, or at least one of $u$'s $L$-hop neighbors ($L$ is the number of layers in RAINDROP's message passing) is recorded. We use $||$ and $^T$ to denote concatenation and transpose, respectively. We omit layer index $l \in \{1, \ldots, L\}$ for simplicity when text is clear.

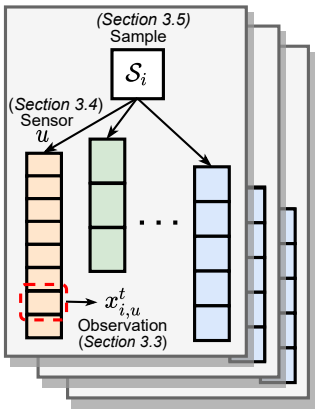



**Figure 2:** Hierarchical structure of irregular multivariate time series dataset. RAINDROP embeds individual observations considering inter-sensor dependencies (Sec. 3.3), aggregates them into a sensor embedding using temporal attention (Sec. 3.4), and finally aggregates sensor embeddings into a sample embedding (Sec. 3.5).

**Problem (Representation learning for irregularly sampled multivariate time series).** *A dataset $\mathcal{D}$ of irregularly sampled multivariate time series is given, where each sample $\mathcal{S}_i$ has multiple sensors and each sensor has a variable number of observations.* RAINDROP *learns a function $f : \mathcal{S}_i \rightarrow \boldsymbol{z}_i$ that maps $\mathcal{S}_i$ to a fixed-length representation $\boldsymbol{z}_i$ suitable for downstream task of interest, such as classification. Using learned $\boldsymbol{z}_i$,* RAINDROP *can predict label $\hat{y}_i \in \{1, \ldots, C\}$ for $\mathcal{S}_i$.*

RAINDROP learns informative embeddings for irregularly samples time series. The learned embeddings capture temporal patterns of irregular observations and explicitly consider varying dependencies between sensors. While we focus on time-series classification in this work, the proposed method can be easily extended to broader applications such as regression, clustering and generation tasks.

### 3.1 OVERVIEW OF RAINDROP

RAINDROP aims to learn a fixed-dimensional embedding $\boldsymbol{z}_i$ for given $\mathcal{S}_i$, and then predict the associated label $\hat{y}_i$. RAINDROP learns sample embeddings in a hierarchical architecture composed of observation embedding, sensor embedding and sample embedding (Figure 2). We describe the procedure of RAINDROP in the context of at most one observation at a specific time $t$ (one sensor has observation while all residual sensors do not have observation). If there are multiple observations at the same time, RAINDROP can effortlessly process them all in parallel.

We first construct a graph for each sample where nodes represent sensors and edges indicate relations between sensors (Sec. 3.2). Each sample $\mathcal{S}_i$ corresponds to a graph $\mathcal{G}_i$, where we use $e_{i,uv}$ to denote the weight of directed edge going from sensor $u$ to sensor $v$. The graph structures are automatically learned while considering sample-wise and time-wise specificity.

The key idea of RAINDROP is to borrow information from $u$'s neighbors based on estimated relationships between $u$ and other sensors. This is achieved via message passing carried out on $\mathcal{S}_i$'s dependency graph and initiated at node $u$ in the graph. When an observation $(t, x_{i,u}^t)$ is recorded for sample $\mathcal{S}_i$ at time $t$, RAINDROP first embeds the observation at *active* sensor $u$ (*i.e.*, sensor whose value was recorded) and then propagates messages (*i.e.*, the observation embeddings) from $u$ to neighboring sensors along edges in sensor dependency graph $\mathcal{G}_i$. As a result, recording the value of $u$ can affect $u$'s embedding as well as embeddings of other sensors that behave similarly as $u$ (Sec. 3.3). Finally, RAINDROP generates sensor embeddings by aggregating all observation embeddings for each sensor (across all timestamps) using temporal attention weights (Sec. 3.4). At last, RAINDROP embeds every sample $\mathcal{S}_i$ based on the embeddings of all sensors (Sec. 3.5) and feeds the learned sample embedding into downstream classifier.

### 3.2 CONSTRUCTING SENSOR DEPENDENCY GRAPHS

We build a directed weighted graph $\mathcal{G}_i = \{\mathcal{V}, \mathcal{E}_i\}$ for every sample $\mathcal{S}_i$ and refer to it as the *sensor dependency graph* for $\mathcal{S}_i$. Nodes $\mathcal{V}$ represent sensors and edges $\mathcal{E}_i$ describe dependencies between sensors in sample $\mathcal{S}_i$ that RAINDROP infers. As we show in experiments, RAINDROP can be directly used with samples that only contain a subset of sensors in $\mathcal{V}$. We denote edge from $u$ to $v$ as a triplet $(u, e_{i,uv}, v)$, where $e_{i,uv} \in [0, 1]$ represents the strength of relationship between sensors $u$ and $v$ in sample $\mathcal{S}_i$. Edge $(u, e_{i,uv}, v)$ describes the relationship between $u$ and $v$: when $u$ receives an observation, it will send a neural message to $v$ following edge $e_{i,uv}$. If $e_{i,uv} = 0$, there is no exchange of neural information between $u$ and $v$, indicating that the two sensors are unrelated. We assume that the importance of $u$ to $v$ is different than the importance of $v$ to $u$, and so we treat sensor dependency graphs as directed, *i.e.*, $e_{i,uv} \neq e_{i,vu}$. All graphs are initialized as fully-connected graphs (*i.e.*, $e_{i,uv} = 1$ for any $u$, $v$ and $\mathcal{S}_i$) and edge weights $e_{i,uv}$ are updated following Eq. 3 during model training. If available, it is easy to integrate additional domain knowledge into graph initialization.

### 3.3 GENERATING EMBEDDINGS OF INDIVIDUAL OBSERVATIONS

Let sensor $u$ be activated at time $t \in \mathcal{T}_{i,u}$ and receive observation $x_{i,u}^t$ and let $u$ be connected to $v$ through edge $(u, e_{i,uv}, v)$. We next describe how to produce observation embeddings $\boldsymbol{h}_{i,u}^t \in \mathbb{R}^{d_h}$ and $\boldsymbol{h}_{i,v}^t \in \mathbb{R}^{d_h}$ for sensors $u$ and $v$, respectively (Figure 3a). We omit layer index $l$ and note that the proposed strategy applies to any number of layers.

**Embedding observation of an active sensor.** We treat $u$ as an active sensor that has received observation $x_{i,u}^t$. For sufficient expressive power (Veličković et al., 2018), we map observation $x_{i,u}^t$ to a high-dimensional space using a nonlinear transformation: $\boldsymbol{h}_{i,u}^t = \sigma(x_{i,u}^t \boldsymbol{R}_u)$. Here, we use sensor-specific transformations because values recorded at different sensors can follow different distributions, which is evidenced by trainable weight vectors $\boldsymbol{R}_u$ depending on what sensor is activated (Li et al., 2020). Alternative functions (such as a multilayer perceptron) can be considered for transforming $x_{i,u}^t$ into $\boldsymbol{h}_{i,u}^t$. As $\boldsymbol{h}_{i,u}^t$ represents information brought on by measuring $x_{i,u}^t$, we regard $\boldsymbol{h}_{i,u}^t$ as $u$'s observation embedding at $t$. The sensor-specific weight vectors $\boldsymbol{R}_u$ are shared across all samples.

**Message passing in sensor dependency graphs.** For sensors that are not active at timestamp $t$ but are neighbors of the active sensor $u$ in the sensor dependency graph $\mathcal{G}_i$, RAINDROP uses relationships between $u$ and those sensors to estimate observation embeddings for them. We proceed by describing

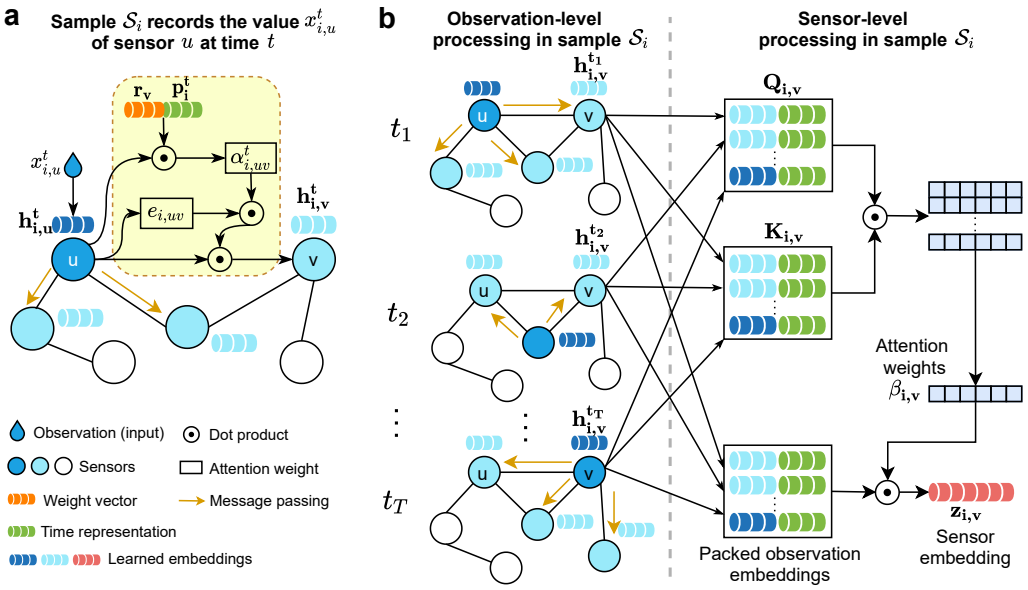



**Figure 3:** **(a)** RAINDROP generates observation embedding $\boldsymbol{h}^t_{i,u}$ based on observed value $x^t_{i,u}$ at $t$, passes message to neighbor sensors such as $v$, and generates $\boldsymbol{h}^t_{i,v}$ through inter-sensor dependencies. The $\alpha^t_{i,uv}$ denotes a time-specific attention weight, calculated based on time representation $\boldsymbol{p}^t_i$ and weight vector $\boldsymbol{r}_v$. Edge weight $e_{i,uv}$ is shared by all timestamps. **(b)** An illustration of generating sensor embedding. Apply the message passing in (a) to all timestamps and produce corresponding observation embeddings. We aggregate arbitrary number of observation embeddings into a fixed-length sensor embedding $\boldsymbol{z}_{i,v}$ while paying distinctive attentions to different observations. We independently apply the processing procedure to all sensors.

how RAINDROP generates observation embedding $\boldsymbol{h}^t_{i,v}$ for sensor $v$ assuming $v$ is a neighbor of $u$ in $\mathcal{G}_i$. Given $\boldsymbol{h}^t_{i,u}$ and edge $(u, e_{i,uv}, v)$, we first calculate inter-sensor attention weight $\alpha^t_{i,uv} \in [0,1]$, representing how important $u$ is to $v$ via the following equation:

$$\alpha^t_{i,uv} = \sigma(\boldsymbol{h}^t_{i,u}\boldsymbol{D}[\boldsymbol{r}_v||\boldsymbol{p}^t_i]^T), \tag{1}$$

where $\boldsymbol{r}_v \in \mathbb{R}^{d_r}$ is a trainable weight vector that is specific to the sensor receiving the message (*i.e.*, $\boldsymbol{h}^t_{i,u}$). Vector $\boldsymbol{r}_v$ allows the model to learn distinct attention weights for different edges going out from the same sensor $u$. Further, $\boldsymbol{p}^t_i \in \mathbb{R}^{d_t}$ is the timestamp encoding obtained by converting a 1-dimensional timestamp $t$ into a multi-dimensional vector $\boldsymbol{p}^t_i$ by passing $t$ through a series of trigonometric functions with varying frequencies (Horn et al., 2020). See Appendix A.1 for details. RAINDROP uses $\boldsymbol{p}^t_i$ to calculate attention weights that are sensitive to time. Finally, $\boldsymbol{D}$ is a trainable weight matrix mapping $\boldsymbol{h}^t_{i,u}$ from a $d_h$ dimensions to $(d_r + d_t)$ dimensions. Taken this together, we can estimate the embedding $\boldsymbol{h}^t_{i,v}$ for $u$'s neighbor $v$ as follows:

$$\boldsymbol{h}^t_{i,v} = \sigma(\boldsymbol{h}^t_{i,u}\boldsymbol{w}_u\boldsymbol{w}_v^T\alpha^t_{i,uv}e_{i,uv}), \tag{2}$$

where $\boldsymbol{w}_u, \boldsymbol{w}_v \in \mathbb{R}^{d_h}$ are trainable weight vectors shared across all samples, $\boldsymbol{w}_u$ is specific to active sensor $u$, and $\boldsymbol{w}_v$ is specific to neighboring sensor $v$. In the above equation, $e_{i,uv}$ denotes edge weight shared across all timestamps. The above message passing describes the processing of a single observation at a single timestamp. In case multiple sensors are active at time $t$ and connected with $v$, we normalize $\alpha^t_{i,uv}$ (with softmax function) across active sensors and aggregate messages at every $v$.

Overall, RAINDROP produces observation embedding $\boldsymbol{h}^t_{i,v}$ for sensor $v$ through its relational connection with $u$, even though there is no direct measurement of $v$ at time $t$. These message passing operations are performed to adaptively and dynamically estimate missing observations in the embedding space based on recorded information and learned graph structure.

**Updating sensor dependency graphs.** We describe the update of edge weights and prune of graph structures in the situation that stacks multiple RAINDROP layers. Here we explicitly show layer index $l$ because multiple layers are involved in the computation. As no prior knowledge is assumed, we initialize the graph as all sensors connected with each other. However, the fully connected edges

may connect sensors that should be independent, which will introduce spurious correlations and prevent the model from paying attention to the truly important connections. Addressing this issue, RAINDROP automatically updates edge weights and prunes out less important edges. Based on the aggregated temporal influence driven by the inter-sensor attention weights $\alpha_{i,uv}^{(l),t}$, we update edge weights $e_{i,uv}^{(l)}$ in each layer $l \in \{1, \ldots, L\}$ by:

$$e_{i,uv}^{(l)} = \frac{e_{i,uv}^{(l-1)}}{|\mathcal{T}_{i,u}|} \sum_{t \in \mathcal{T}_{i,u}} \alpha_{i,uv}^{(l),t}, \tag{3}$$

where $\mathcal{T}_{i,u}$ denotes the set of all timestamps where there is message passes from $u$ to $v$. In particular, we set $e_{i,uv}^{(0)} = 1$ in the initialization of graph structures. We use $L = 2$ in all our experiments. In every layer, we order the estimated values $e_{i,uv}^{(l)}$ for all edges in sample $\mathcal{S}_i$ and prune bottom $K\%$ edges with smallest edge weights (Yang et al., 2021). Pruned edges will not re-appear in later layers.

### 3.4 GENERATING SENSOR EMBEDDINGS

Next we describe how to aggregate observation embeddings into sensor embeddings, taking sensor $v$ as an example (Figure 3b). Previous step outlined in Sec. 3.3 outputs observation embeddings for every timestamp when either $v$ or $v$'s neighbor in $\mathcal{S}_i$'s sensor dependency graph are observed. The following step aggregates those embeddings into a sensor embedding $\boldsymbol{z}_{i,v}$ via temporal attention.

We concatenate observation embedding $\boldsymbol{h}_{i,v}^t$ with time representation $\boldsymbol{p}_i^t$ to include information of timestamp. Then, we pack the concatenated embeddings $[\boldsymbol{h}_{i,v}^t || \boldsymbol{p}_i^t]$ for all $t \in \mathcal{T}_{i,v}$ into a matrix $\boldsymbol{H}_{i,v}$. The $\mathcal{T}_{i,v} = \{t_1, t_2, \ldots, t_T\}$ includes all timestamps when a readout is observed in $v$ (we can directly generate $\boldsymbol{h}_{i,v}^t$) or $v$'s neighbor (we can generate $\boldsymbol{h}_{i,v}^t$ through message passing). We calculate the temporal attention weight $\beta_{i,v}^t$ using self-attention (Hu et al., 2020; Yun et al., 2019; Vaswani et al., 2017). The $\beta_{i,v}^t$ represents the importance of observation embedding to the whole sensor embedding $\boldsymbol{z}_{i,v}$. The $\beta_{i,v}^t$ is the corresponding element of $\beta_{i,v}$ that is calculated through:

$$\beta_{i,v} = \text{softmax}\left(\frac{\boldsymbol{Q}_{i,v}\boldsymbol{K}_{i,v}^T}{\sqrt{d_k}}\boldsymbol{s}\right), \tag{4}$$

where query matrix $\boldsymbol{Q}_{i,v} = \boldsymbol{H}_{i,v}\boldsymbol{W}_Q$ and key matrix $\boldsymbol{K}_{i,v} = \boldsymbol{H}_{i,v}\boldsymbol{W}_K$ are two matrices linearly mapped from $\boldsymbol{H}_{i,v}$ parameterized by $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$, respectively. The $\sqrt{d_k}$ is a scalar factor where $d_k$ is the dimension after linear mapping. Our calculation of $\beta_{i,v}$ is different from the typical self-attention by having mapping weight vector $\boldsymbol{s}$. The standard dot-product self-attention generates an attention matrix with dimension of $T \times T$ (where $T = |\mathcal{T}_{i,v}|$ can vary across samples) that has an attention weight for each pair of observation embeddings. In our case, we only need a single attention vector where each element denotes the attention we should pay to an observation embedding when generating the sensor embedding. Thus, we modify the self-attention model to fit our case and use a trainable $\boldsymbol{s}$ to map the results of dot-product from matrix ($\mathbb{R}^{T \times T}$) to vector ($\mathbb{R}^T$).

We calculate sensor embedding $\boldsymbol{z}_{i,v}$ through:

$$\boldsymbol{z}_{i,v} = \sum_{t \in \mathcal{T}_{i,v}} (\beta_{i,v}^t [\boldsymbol{h}_{i,v}^t || \boldsymbol{p}_i^t] \boldsymbol{W}), \tag{5}$$

where weight matrix $\boldsymbol{W}$ is a linear projector shared by all sensors and samples. It's worth to mention that all attention weights (such as $\alpha_{i,uv}^t$ and $\beta_{i,v}$) can be multi-head. In this work, we describe the model using a single head for brevity.

Using attentional aggregation, RAINDROP can learn a fixed-length sensor embedding for arbitrary number of observations. Meanwhile, RAINDROP is capable of focusing on the most informative observation embeddings. We process all observation embeddings as a whole instead of sequentially, which allows parallel computation for faster training and also mitigates the performance drop caused by modeling long dependencies sequentially. In the case of sensors with very large number of observations, we can reduce the length of time series by subsampling or splitting a long series into multiple short series.

### 3.5 GENERATING SAMPLE EMBEDDINGS

Finally, we aggregate sensor embeddings $\boldsymbol{z}_{i,v}$ (Eq. 5) to obtain an embedding $\boldsymbol{z}_i \in \mathbb{R}^{d_z}$ for sample $\mathcal{S}_i$, as follows: $\boldsymbol{z}_i = [\boldsymbol{z}_{i,1}||\boldsymbol{z}_{i,2}||\cdots||\boldsymbol{z}_{i,M}]$ across all sensors $v = 1, 2, \ldots, M$. Our preliminary experiments show that concatenation outperforms other popular aggregation functions (*e.g.*, average (Errica et al., 2021) and squeeze-excitation readout function (Kim et al., 2021; Hu et al., 2018)). While any of those aggregation functions can be considered, we used concatenation throughout all experiments in this manuscript. Taken together Sec. 3.2-3.5, given an input of sample $\mathcal{S}_i$, RAINDROP returns a sample embedding $\boldsymbol{z}_i$ that can in turn be fed into a downstream task.

### 3.6 IMPLEMENTATION AND PRACTICAL CONSIDERATIONS

**Loss function.** The RAINDROP's loss function is formulated as: $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_r$, where $\mathcal{L}_r = \frac{1}{M^2}\sum_{u,v\in\mathcal{V}}\sum_{i,j\in\mathcal{D}}||e_{i,uv} - e_{j,uv}||_2/(N-1)^2$, $\mathcal{L}_{\text{CE}}$ is a cross-entropy classification loss, and $\mathcal{L}_r$ is a regularizer that encourages the model to learn similar sensor dependency graphs for similar samples. The $\mathcal{L}_r$ measures averaged Euclidean distance of edge weights across all samples pairs, in all sensor pairs (including self-connections). Practically, as $N$ can be large, we calculate $\mathcal{L}_r$ only for samples in each batch.

**Downstream tasks.** If a sample has auxiliary attributes (*e.g.*, a patient's demographics) that do not change over time, we can project the attribute vector to a $d_a$-dimensional vector $\boldsymbol{a}_i$ with a fully-connected layer and concatenate it with the sample embedding, getting $[\boldsymbol{z}_i||\boldsymbol{a}_i]$. At last, we feed $[\boldsymbol{z}_i||\boldsymbol{a}_i]$ (or only $\boldsymbol{z_i}$ if $\boldsymbol{a}_i$ is not available) into a neural classifier $\varphi : \mathbb{R}^{d_z+d_a} \to \{1, \ldots, C\}$. In our experiments, $\varphi$ is a 2-layer fully-connected network with $C$ neurons at the output layer returning prediction $\hat{y}_i = \varphi([\boldsymbol{z}_i||\boldsymbol{a}_i])$ for sample $\mathcal{S}_i$.

**Sensor dependencies.** While modeling sensor dependencies, we involve observation embedding ($\boldsymbol{h}_{i,u}^t$, Eq. 1) of each sample in the calculation of attention weights. Similarly, to model time-wise specificity in graph structures, we consider time information ($\boldsymbol{p}_i^t$, Eq. 1) when measuring $\alpha_{i,uv}^t$. RAINDROP can capture similar graph structures across samples from three aspects: (1) the initial graphs are the same in all samples; (2) the parameters in message passing ($\boldsymbol{R}_u$; $\boldsymbol{w}_u$, $\boldsymbol{w}_v$, Eq. 2), inter-sensor attention weights calculation ($\boldsymbol{D}$, Eq. 1), and temporal attention weights calculation ($\boldsymbol{W}$, Eq. 5; $\boldsymbol{s}$, Eq. 4) are shared by all samples; (3) we encourage the model to learn similar graph structures by adding a penalty to disparity of structures ($\mathcal{L}_r$).

**Scalability.** RAINDROP is efficient because embeddings can be learned in parallel. In particular, processing of observation embeddings is independent across timestamps. Similarly, sensor embeddings can be processed independently across different sensors (Figure 3). While the complexity of temporal self-attention calculation grows quadratically with the number of observations (Eq. 4), it can be practically implemented using highly-optimized matrix multiplication.

## 4 EXPERIMENTS

**Datasets.** Below we briefly overview healthcare and human activity datasets. (1) **P19** (Reyna et al., 2020) includes 38,803 patients that are monitored by 34 sensors. Each patient is associated with a binary label representing the occurrence of sepsis. (2) **P12** (Goldberger et al., 2000) records temporal measurements of 36 sensors of 11,988 patients in the first 48-hour stay in ICU. The samples are labeled based on hospitalization length. (3) **PAM** (Reiss & Stricker, 2012) contains 5,333 segments from 8 activities of daily living that are measured by 17 sensors. Details are in Appendix A.2.

**Baselines.** We compare RAINDROP with five state-of-the-art time series classification methods: *Transformer* (Vaswani et al., 2017), *Trans-mean*, *GRU-D* (Che et al., 2018), *SeFT* (Horn et al., 2020), and *mTAND* (Shukla & Marlin, 2021). The *Trans-mean* is an imputation method combining transformer architecture with commonly used average interpolation (*i.e.*, missing values are replaced by average observations in each sensor). Further details are in Section 2. The *mTAND* (Shukla & Marlin, 2021) method has been shown to outperform numerous recurrent models including *RNN-Impute* (Che et al., 2018), *RNN-Simple*, *Phased-LSTM* (Neil et al., 2016), and *IP-Nets* (Shukla & Marlin, 2018), along with ordinary differential equations (ODE)-based models such as *LATENT-ODE* and *ODE-RNN* (Chen et al., 2018). For this reason, we compare with *mTAND* and do not report

comparison with those techniques in this paper. Details on hyperparameter selection of RAINDROP and baselines are in Appendix A.4. Chosen evaluation metrics are presented in Appendix A.3.

### 4.1 RESULTS ACROSS A MULTITUDE OF EVALUATION SETTINGS

**Setting 1: Classic time series classification.** (1) Setup. We randomly split the dataset into training (80%), validation (10%), and test (10%) set. The indices of these splits are fixed across all methods. (2) Results. As shown in Table 1, RAINDROP obtains the best performance across three benchmark datasets, implying our superiority in standard time series classification scenario. In particular, in binary classification (P19 and P12), RAINDROP outperforms the strongest baselines by 5.3% in AUROC and 4.8% in AUPRC on average. In a much more challenging task of 8-class classification, RAINDROP also outperforms existing approaches by 5.7% in accuracy and 5.5% in F1 score on PAM.

**Table 1:** Method benchmarking on irregularly sampled time series classification task (Setting 1).

| Models | P19 | | P12 | | PAM | | | |
|---|---|---|---|---|---|---|---|---|
| | AUROC | AUPRC | AUROC | AUPRC | Accuracy | Precision | Recall | F1 score |
| Transformer | 83.2 ± 1.3 | 47.6 ± 3.8 | 65.1 ± 5.6 | 95.7 ± 1.6 | 83.5 ± 1.5 | 84.8 ± 1.5 | 86.0 ± 1.2 | 85.0 ± 1.3 |
| Trans-mean | 84.1 ± 1.7 | 47.4 ± 1.4 | 66.8 ± 4.2 | 95.9 ± 1.1 | 83.7 ± 2.3 | 84.9 ± 2.6 | 86.4 ± 2.1 | 85.1 ± 2.4 |
| GRU-D | 83.9 ±1.7 | 46.9 ± 2.1 | 67.2 ± 3.6 | 95.9 ± 2.1 | 83.3 ± 1.6 | 84.6 ± 1.2 | 85.2 ± 1.6 | 84.8 ± 1.2 |
| SeFT | 78.7 ± 2.4 | 31.1 ± 2.8 | 66.8 ± 0.8 | 96.2 ± 0.2 | 67.1 ± 2.2 | 70.0 ± 2.4 | 68.2 ± 1.5 | 68.5 ± 1.8 |
| mTAND | 80.4 ± 1.3 | 32.4 ± 1.8 | 65.3 ± 1.7 | 96.5 ± 1.2 | 74.6 ± 4.3 | 74.3 ± 4.0 | 79.5 ± 2.8 | 76.8 ± 3.4 |
| RAINDROP | **87.0 ± 2.3** | **51.8 ± 5.5** | **72.1 ± 1.3** | **97.0 ± 0.4** | **88.5 ± 1.5** | **89.9 ± 1.5** | **89.9 ± 0.6** | **89.8 ± 1.0** |

**Setting 2: Leave-fixed-sensors-out.** (1) Setup. We evaluate whether RAINDROP can learn dependencies among sensors, thus achieve good performance even some sensors are completely missing. This setting is practically meaningful, such as facing sensor failure. or some sensors are unavailable in specific scenes. Our intuition is that RAINDROP can compensate for the missing information from nearby observations by exploiting relational dependencies. In this setting, we select a proportion of sensors, and set all their observations as zero in validation and testing set (training samples are not changed). We mask out the most informative sensors which are selected through information gain (Appendix A.5). The selected sensors are fixed across samples and models. (2) Results. We report results taking PAM as an example. In Table 2 (left columns), we observe that RAINDROP achieves the highest values in 18 out of 20 settings while the missing data ratio ranges from 10% to 50%. The missing data ratio refers to the proportion of sensors we mask out. With the increase of missing data ratio, our model has greater relative gains. RAINDROP outperforms competitive baselines by up to 24.9% in accuracy, 50.3% in precision, 29.3% in recall, and 42.8% in F1 score.

**Setting 3: Leave-random-sensors-out.** (1) Setup. Setting 3 is similar to Setting 2 except that the missing sensors in this setting are randomly selected instead of fixed. In each test sample, we select a subset of sensors and regard them as missing through replacing all of their observations with zeros. (2) Results. We provide results of PAM in Table 2 (right columns). Similar to Setting 2, we observe that RAINDROP achieves better performance than baselines in 16 out of 20 settings while Trans-mean and GRU-D are competitive methods.

**Setting 4: Group-wise time series classification.** (1) Setup. To understand whether RAINDROP can adaptively adjust its structure and generalize well to other groups of samples which were not observed while training the model. In this setting we split the data into two groups, based on a specific static attribute. The first split attribute is *age*, where we classify people into *young* ($< 65$ years) and *old* ($\geq 65$ years) groups. We also split patients into *male* and *female* by *gender* attribute. Given the split attribute, we use one group as a train set and randomly split the other group into equally sized validation and test set. (2) Results. Taking P19 as an example, we present the classification results when the training and testing samples are from different groups. As shown in Table 3, RAINDROP achieves the best results over all of the four given cross-group scenarios. For instance, RAINDROP claims large margins (with 4.8% in AUROC and 13.1% in AUPRC absolute improvement) over the second best model while training on males and testing on female patients.

### 4.2 ABLATION STUDY AND VISUALIZATION OF LEARNED GRAPH STRUCTURE

**Ablation study.** Taking PAM in the most common task (Setting 1) as an example, we conduct an ablation study to evaluate the necessity of three fundamental steps of RAINDROP: inter-sensor

**Table 2:** Classification on samples with fixed (Setting 2) and random (Setting 3) missing sensors (PAM dataset). Results for P19 dataset (Settings 2-3) are presented in Appendix A.6.

| Missing data ratio | Models | PAM (Setting 2: leave-**fixed**-sensors-out) | | | | PAM (Setting 3: leave-**random**-sensors-out) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Accuracy | Precision | Recall | F1 score | Accuracy | Precision | Recall | F1 score |
| 10% | Transformer | 60.3 ± 2.4 | 57.8 ± 9.3 | 59.8 ± 5.4 | 57.2 ± 8.0 | 60.9 ± 12.8 | 58.4 ± 18.4 | 59.1 ± 16.2 | 56.9 ± 18.9 |
| | Trans-mean | 60.4 ± 11.2 | 61.8 ± 14.9 | 60.2 ± 13.8 | 58.0 ± 15.2 | 62.4 ± 3.5 | 59.6 ± 7.2 | 63.7 ± 8.1 | 62.7 ± 6.4 |
| | GRU-D | 65.4 ± 1.7 | 72.6 ± 2.6 | 64.3 ± 5.3 | 63.6 ± 0.4 | 68.4 ± 3.7 | 74.2 ± 3.0 | 70.8 ± 4.2 | 72.0 ± 3.7 |
| | SeFT | 58.9 ± 2.3 | 62.5 ± 1.8 | 59.6 ± 2.6 | 59.6 ± 2.6 | 40.0 ± 1.9 | 40.8 ± 3.2 | 41.0 ± 0.7 | 39.9 ± 1.5 |
| | mTAND | 58.8 ± 2.7 | 59.5 ± 5.3 | 64.4 ± 2.9 | 61.8 ± 4.1 | 53.4 ± 2.0 | 54.8 ± 2.7 | 57.0 ± 1.9 | 55.9 ± 2.2 |
| | RAINDROP | **77.2 ± 2.1** | **82.3 ± 1.1** | **78.4 ± 1.9** | **75.2 ± 3.1** | **76.7 ± 1.8** | **79.9 ± 1.7** | **77.9 ± 2.3** | **78.6 ± 1.8** |
| 20% | Transformer | 63.1 ± 7.6 | 71.1 ± 7.1 | 62.2 ± 8.2 | 63.2 ± 8.7 | 62.3 ± 11.5 | 65.9 ± 12.7 | 61.4 ± 13.9 | 61.8 ± 15.6 |
| | Trans-mean | 61.2 ± 3.0 | **74.2 ± 1.8** | 63.5 ± 4.4 | 64.1 ± 4.1 | 56.8 ± 4.1 | 59.4 ± 3.4 | 53.2 ± 3.9 | 55.3 ± 3.5 |
| | GRU-D | 64.6 ± 1.8 | 73.3 ± 3.6 | 63.5 ± 4.6 | 64.8 ± 3.6 | 64.8 ± 0.4 | 69.8 ± 0.8 | 65.8 ± 0.5 | 67.2 ± 0.0 |
| | SeFT | 35.7 ± 0.5 | 42.1 ± 4.8 | 38.1 ± 1.3 | 35.0 ± 2.2 | 34.2 ± 2.8 | 34.9 ± 5.2 | 34.6 ± 2.1 | 33.3 ± 2.7 |
| | mTAND | 33.2 ± 5.0 | 36.9 ± 3.7 | 37.7 ± 3.7 | 37.3 ± 3.4 | 45.6 ± 1.6 | 49.2 ± 2.1 | 49.0 ± 1.6 | 49.0 ± 1.0 |
| | RAINDROP | **66.5 ± 4.0** | 72.0 ± 3.9 | **67.9 ± 5.8** | **65.1 ± 7.0** | **71.3 ± 2.5** | **75.8 ± 2.2** | **72.5 ± 2.0** | **73.4 ± 2.1** |
| 30% | Transformer | 31.6 ± 10.0 | 26.4 ± 9.7 | 24.0 ± 10.0 | 19.0 ± 12.8 | 52.0 ± 11.9 | 55.2 ± 15.3 | 50.1 ± 13.3 | 48.4 ± 18.2 |
| | Trans-mean | 42.5 ± 8.6 | 45.3 ± 9.6 | 37.0 ± 7.9 | 33.9 ± 8.2 | **65.1 ± 1.9** | 63.8 ± 1.2 | **67.9 ± 1.8** | **64.9 ± 1.7** |
| | GRU-D | 45.1 ± 2.9 | 51.7 ± 6.2 | 42.1 ± 6.6 | 47.2 ± 3.9 | 58.0 ± 2.0 | 63.2 ± 1.7 | 58.2 ± 3.1 | 59.3 ± 3.5 |
| | SeFT | 32.7 ± 2.3 | 27.9 ± 2.4 | 34.5 ± 3.0 | 28.0 ± 1.4 | 31.7 ± 1.5 | 31.0 ± 2.7 | 32.0 ± 1.2 | 28.0 ± 1.6 |
| | mTAND | 27.5 ± 4.5 | 31.2 ± 7.3 | 30.6 ± 4.0 | 30.8 ± 5.6 | 34.7 ± 5.5 | 43.4 ± 4.0 | 36.3 ± 4.7 | 39.5 ± 4.4 |
| | RAINDROP | **52.4 ± 2.8** | **60.9 ± 3.8** | **51.3 ± 7.1** | **48.4 ± 1.8** | 60.3 ± 3.5 | **68.1 ± 3.1** | 60.3 ± 3.6 | 61.9 ± 3.9 |
| 40% | Transformer | 23.0 ± 3.5 | 7.4 ± 6.0 | 14.5 ± 2.6 | 6.9 ± 2.6 | 43.8 ± 14.0 | 44.6 ± 23.0 | 40.5 ± 15.9 | 40.2 ± 20.1 |
| | Trans-mean | 25.7 ± 2.5 | 9.1 ± 2.3 | 18.5 ± 1.4 | 9.9 ± 1.1 | 48.7 ± 2.7 | 55.8 ± 2.6 | 54.2 ± 3.0 | 55.1 ± 2.9 |
| | GRU-D | 46.4 ± 2.5 | **64.5 ± 6.8** | 42.6 ± 7.4 | 44.3 ± 7.9 | 47.7 ± 1.4 | 63.4 ± 1.6 | 44.5 ± 0.5 | 47.5 ± 0.0 |
| | SeFT | 26.3 ± 0.9 | 29.9 ± 4.5 | 27.3 ± 1.6 | 22.3 ± 1.9 | 26.8 ± 2.6 | 24.1 ± 3.4 | 28.0 ± 1.2 | 23.3 ± 3.0 |
| | mTAND | 19.4 ± 4.5 | 15.1 ± 4.4 | 20.2 ± 3.8 | 17.0 ± 3.4 | 23.7 ± 1.0 | 33.9 ± 6.5 | 26.4 ± 1.6 | 29.3 ± 1.9 |
| | RAINDROP | **52.5 ± 3.7** | 53.4 ± 5.6 | **48.6 ± 1.9** | **44.7 ± 3.4** | **57.0 ± 3.1** | **65.4 ± 2.7** | **56.7 ± 3.1** | **58.9 ± 2.5** |
| 50% | Transformer | 21.4 ± 1.8 | 2.7 ± 0.2 | 12.5 ± 0.4 | 4.4 ± 0.3 | 43.2 ± 2.5 | 52.0 ± 2.5 | 36.9 ± 3.1 | 41.9 ± 3.2 |
| | Trans-mean | 21.3 ± 1.6 | 2.8 ± 0.4 | 12.5 ± 0.7 | 4.6 ± 0.2 | 46.4 ± 1.4 | 59.1 ± 3.2 | 43.1 ± 2.2 | 46.5 ± 3.1 |
| | GRU-D | 37.3 ± 2.7 | 29.6 ± 5.9 | 32.8 ± 4.6 | 26.6 ± 5.9 | **49.7 ± 1.2** | 52.4 ± 0.3 | 42.5 ± 1.7 | 47.5 ± 1.2 |
| | SeFT | 24.7 ± 1.7 | 15.9 ± 2.7 | 25.3 ± 2.6 | 18.2 ± 2.4 | 26.4 ± 1.4 | 23.0 ± 2.9 | 27.5 ± 0.4 | 23.5 ± 1.8 |
| | mTAND | 16.9 ± 3.1 | 12.6 ± 5.5 | 17.0 ± 1.6 | 13.9 ± 4.0 | 20.9 ± 3.1 | 35.1 ± 6.1 | 23.0 ± 3.2 | 27.7 ± 3.9 |
| | RAINDROP | **46.6 ± 2.6** | **44.5 ± 2.6** | **42.4 ± 3.9** | **38.0 ± 4.0** | 47.2 ± 4.4 | **59.4 ± 3.9** | **44.8 ± 5.3** | **47.6 ± 5.2** |

**Table 3:** Classification results when train and test samples originate from different groups (P19; Setting 4).

| Model | Generalizing to a new patient group | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Train: Young → Test: Old | | Train: Old → Test: Young | | Train: Male → Test: Female | | Train: Female → Test: Male | |
| | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC |
| Transformer | 76.2 ± 0.7 | 30.5 ± 4.8 | 76.5 ± 1.1 | 33.7 ± 5.7 | 77.8 ± 1.1 | 26.0 ± 6.2 | 75.2 ± 1.0 | 30.3 ± 5.5 |
| Trans-mean | 80.6 ± 1.4 | 39.8 ± 4.2 | 78.4 ± 1.1 | 35.8 ± 2.9 | 80.2 ± 1.7 | 32.1 ± 1.9 | 76.4 ± 0.8 | 32.5 ± 3.3 |
| GRU-D | 76.5 ± 1.7 | 29.5 ± 2.3 | 79.6 ± 1.7 | 35.2 ± 4.6 | 78.5 ± 1.6 | 31.9 ± 4.8 | 76.3 ± 2.5 | 31.1 ± 2.6 |
| SeFT | 77.5 ± 0.7 | 26.6 ± 1.2 | 78.9 ± 1.0 | 32.7 ± 2.7 | 78.6 ± 0.6 | 31.1 ± 1.2 | 76.9 ± 0.5 | 26.4 ± 1.1 |
| mTAND | 79.0 ± 0.8 | 28.8 ± 2.3 | 79.4 ± 0.6 | 29.8 ± 1.2 | 78.0 ± 0.9 | 26.5 ± 1.7 | 78.9 ± 1.2 | 29.2 ± 2.0 |
| RAINDROP | **83.2 ± 1.6** | **43.6 ± 4.7** | **82.0 ± 4.4** | **44.3 ± 3.6** | **85.0 ± 1.4** | **45.2 ± 2.9** | **81.2 ± 3.8** | **40.7 ± 2.9** |

dependency (further decomposed to key weights including $e_{i,uv}$, $\boldsymbol{r}_v$, $\boldsymbol{p}_i^t$, and $\alpha_{i,uv}^t$), temporal attention, and sensor-level concatenation. In Appendix A.7 (Table 6), we observe that all of the components and the regularization term $\mathcal{L}_r$ are important to improve model performance.

**Visualizing learned sensor dependency graphs.** We investigate whether similar samples (*i.e.*, with same label) have similar graph structures by visualizing the learned inter-sensor dependencies (P19; Setting 1). As shown in Figure 4, we can observe distinguishable patterns between the graph structures of negative and positive samples, indicating RAINDROP can adaptively learn relationships that are sensitive to the classification task. The differential analysis can also provide inspirations to domain experts such as intensivists. More details in Appendix A.8.

## 5 CONCLUSION

We develop RAINDROP, a graph-guided network for learning with irregularly sampled multivariate time series. RAINDROP learns a distinct sensor dependency graph for every sample that captures time-varying dependencies between sensors in the sample. The ability to leverage the relational structure gives RAINDROP unique ability to naturally handle misaligned observations, non-uniform time intervals between successive observations, and sensors with a varying number of recorded observations. Results on three datasets and four experimental settings show that RAINDROP outperforms state-of-the-art methods on multiclass and binary classification tasks, captures evolving relationships between sensors and how they map to labels. Our novel findings have implications for using message passing as a powerful approach to take advantage of relational structure in multivariate time series.

## REPRODUCIBILITY STATEMENT

We ensure the reproducibility of our work by clearly presenting the model and provide publicly accessible code and data. For all datasets used in this work, we share the downloadable links to the raw sources, and well-processed and ready-to-run datasets, with the research community through link https://anonymous.4open.science/r/Raindrop. We specify all the training details (e.g., preprocessing, data splits, hyperparameters, sensor selection) in the main text and Appendix. Python implementation of RAINDROP and all baseline methods is available at the aforementioned link. Detailed description of all data and scripts, configurations, along with examples of usage, will also be provided.

## ETHICS STATEMENT

The ability of RAINDROP model to learn powerful information about sensors' representations and dependencies creates new opportunities for applications, where time series data is predominant, *e.g.*, in healthcare, biology, and finance. In all these fields, especially in patient care, our method should be used with caution. Although our model can gain useful insights from time series data, users must be aware of the restrictions in terms of machine-guided predictions. The use of our model in real-life clinical settings should always be verified by qualified physicians. As with all data-driven solutions, there exists an opportunity for bias in our predictions. In the case of biomedical data, biases can exist within the data itself, which can be, for example, caused by considering static attributes, such as age, weight, and gender. When the ratio of target labels is highly imbalanced, our model mitigates the possible bias by performing minority class upsampling for every processed batch.

All datasets in this paper are publicly available and are not associated with any privacy or security concerns. Further, all data are anonymized to guard against breaching patient's protected health information (PHI). We have followed PhysioNet privacy policy and guidelines (https://archive.physionet.org/privacy.shtml) when experimenting with P12 and P19 datasets.

## REFERENCES


Amaia Abanda, Usue Mori, and Jose A Lozano. A review on distance based time series classification. Data Mining and Knowledge Discovery, 33(2):378–412, 2019.

Denis Agniel, Isaac S Kohane, and Griffin M Weber. Biases in electronic health record data due to processes within the healthcare system: retrospective observational study. British Medical Journal, 361, 2018.

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

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In NeurIPS, 2018.

Zekai Chen, E Jiaze, Xiao Zhang, Hao Sheng, and Xiuzheng Cheng. Multi-task time series forecasting with shared attention. In ICDM Workshop, pp. 917–925, 2020.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014.

Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew Dai. Learning the graphical structure of electronic health records with graph convolutional transformer. In AAAI, volume 34, pp. 606–613, 2020.

Federico Errica, Davide Bacciu, and Alessio Micheli. Graph mixture density networks. In ICML, pp. 3025–3035, 2021.

Chenguang Fang and Chen Wang. Time series data imputation: A survey on deep learning approaches. arXiv:2011.11347, 2020.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 33(4):917–963, 2019.

Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. arXiv:2006.12179, 2020.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. arXiv:2009.10847, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In ICML, pp. 1263–1272. PMLR, 2017.

Ary L. Goldberger, Luis A. Nunes Amaral, L Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and Harry Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circulation, 101 23:E215–20, 2000.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9(8): 1735–1780, 1997.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. pp. 4303–4313, 2020.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In CVPR, pp. 7132–7141, 2018.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In The Web Conference, pp. 2704–2710, 2020.

Ekaterina Kalinicheva, Dino Ienco, Jérémie Sublime, and Maria Trocan. Unsupervised change detection analysis in satellite image time series using deep learning combined with graph-based approaches. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13:1450–1466, 2020.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. arXiv:2005.08926, 2020.

Byung-Hoon Kim, Jong Chul Ye, and Jae-Jin Kim. Learning dynamic graph representation of brain connectome with spatio-temporal attention. arXiv:2105.13495, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.

Michelle M Li, Kexin Huang, and Marinka Zitnik. Representation learning for networks in biology and medicine: Advancements, challenges, and opportunities. arXiv:2104.04883, 2021.

S. C.-X. Li and B. M. Marlin. A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In NIPS, pp. 1804–1812, 2016.

Steven Cheng-Xian Li and Benjamin Marlin. Learning from irregularly-sampled time series: A missing data perspective. In ICML, pp. 5937–5946. PMLR, 2020.

Xiaoxue Li, Yanmin Shang, Yanan Cao, Yangxi Li, Jianlong Tan, and Yanbing Liu. Type-aware anchor link prediction across heterogeneous networks based on graph attention network. In AAAI, volume 34, pp. 147–155, 2020.

Wu Lin, Nicolas Hubacher, and Mohammad Emtiyaz Khan. Variational message passing with structured inference networks. ICLR, 2018.

R. J. Little and D. B. Rubin. Statistical Analysis with Missing Data. John Wiley & Sons, 3 edition, 2014.

Qianli Ma, Sen Li, and Garrison Cottrell. Adversarial joint-learning recurrent neural network for incomplete time series classification. TPAMI, 2020.

Karl Øyvind Mikalsen, Cristina Soguero-Ruiz, Filippo Maria Bianchi, Arthur Revhaug, and Robert Jenssen. Time series cluster kernels to exploit informative missingness and incomplete label information. Pattern Recognition, 115:107896, 2021.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: accelerating recurrent network training for long or event-based sequences. In NIPS, pp. 3889–3897, 2016.

Giannis Nikolentzos, Antoine Tixier, and Michalis Vazirgiannis. Message passing attention networks for document understanding. In AAAI, volume 34, pp. 8544–8551, 2020.

S Ravuri, K Lenc, M Willson, D Kangin, R Lam, P Mirowski, M Athanassiadou, S Kashem, S Madge, R Prudden, et al. Skillful precipitation nowcasting using deep generative models of radar, arxiv. Nature, 597:672–677, 2021.

Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In ISWC, pp. 108–109, 2012.

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, 2020.

Pau Riba, Andreas Fischer, Josep Lladós, and Alicia Fornés. Learning graph distances with message passing neural networks. In ICPR, pp. 2239–2244. IEEE, 2018.

J. L. Schafer and J. W. Graham. Missing data: Our view of the state of the art. Psychological Methods, 7(2), 2002.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied Soft Computing, 90:106181, 2020.

Siyuan Shan and Junier B Oliva. Nrtsi: Non-recurrent time series imputation for irregularly-sampled data. arXiv:2102.03340, 2021.

Satya Narayan Shukla and Benjamin Marlin. Interpolation-prediction networks for irregularly sampled time series. In ICLR, 2018.

Satya Narayan Shukla and Benjamin Marlin. Multi-time attention networks for irregularly sampled time series. In ICLR, 2021.

Satya Narayan Shukla and Benjamin M Marlin. A survey on principles, models and methods for learning from irregularly sampled time series. 2020.

Rafael T Sousa, Lucas A Pereira, and Anderson S Soares. Improving irregularly sampled time series learning with dense descriptors of time. arXiv:2003.09291, 2020.

Qingxiong Tan, Mang Ye, Baoyao Yang, Siqi Liu, Andy Jinhua Ma, Terry Cheuk-Fung Yip, Grace Lai-Hung Wong, and PongChi Yuen. DATA-GRU: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. In AAAI, volume 34, pp. 930–937, 2020.

MingJian Tang, Mamoun Alazab, Yuxiu Luo, and Matthew Donlon. Disclosure of cyber security vulnerabilities: time series modelling. International Journal of Electronic Security and Digital Forensics, 10(3):255–275, 2018.

Sindhu Tipirneni and Chandan K Reddy. Self-supervised transformer for multivariate clinical time-series with missing values. arXiv:2107.14293, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, pp. 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In ICLR, 2018.

Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial temporal graph neural network. In The Web Conference 2020, pp. 1082–1092, 2020.

B. J. Wells, K. M. Chagin, A. S. Nowacki, and M. W. Kattan. Strategies for handling missing data in electronic health record derived data. EGEMS, 1(3), 2013.

Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. volume 33, 2020a.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4–24, 2020b.

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 KDD, pp. 753–763, 2020c.

Jianing Yang, Yongxin Wang, Ruitao Yi, Yuying Zhu, Azaan Rehman, Amir Zadeh, Soujanya Poria, and Louis-Philippe Morency. Mtag: Modal-temporal attention graph for unaligned human multimodal language sequences. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1009–1021, 2021.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. volume 32, pp. 11983–11993, 2019.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In KDD, pp. 2114–2124, 2021.

Li Zhang, Dan Xu, Anurag Arnab, and Philip HS Torr. Dynamic graph message passing networks. In CVPR, pp. 3726–3735, 2020.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. AI Open, 1:57–81, 2020.

# A APPENDIX

## A.1 TIMESTAMP ENCODING

For a given time value $t$, we pass it to trigonometric functions with the frequency of 10,000 (Vaswani et al., 2017) and generate time representation $\boldsymbol{p}^t \in \mathbf{R}^\xi$ (omit sample index $i$ for brevity) through (Horn et al., 2020):

$$\boldsymbol{p}^t_{2k} = \sin(\frac{t}{t^{2k/\xi}}), \quad \boldsymbol{p}^t_{2k+1} = \cos(\frac{t}{t^{2k/\xi}}), \tag{6}$$

where $\xi$ is the expected dimension. In this work, we set $\xi = 16$ in all experimental settings for all models. Please note, we encode the *time value* which is a continuous timestamp, instead of *time position* which is a discrete integer indicating the order of observation in time series.

## A.2 FURTHER DETAILS ON DATASETS

**P19: PhysioNet Sepsis Early Prediction Challenge 2019.** P19 dataset (Reyna et al., 2020) contains 38,803 patients and each patient is monitored by 34 irregularly sampled sensors including 8 vital signs and 26 laboratory values. The original dataset has 40,336 patients, we remove the samples with too short or too long time series, remaining 38,803 patients (the longest time series of the patient has more than one and less than 60 observations). Each patient is associated with a static vector indicating attributes: age, gender, time between hospital admission and ICU admission, ICU type, and ICU length of stay (days). Each patient has a binary label representing occurrence of sepsis within the next 6 hours. The dataset is highly imbalanced with only $\sim$4% positive samples.

**P12: PhysioNet Mortality Prediction Challenge 2012.** P12 dataset (Goldberger et al., 2000) includes 11,988 patients (samples), after removing 12 inappropriate samples following (Horn et al., 2020). Each patient contains multivariate time series with 36 sensors (excluding weight), which are collected in the first 48-hour stay in ICU. Each sample has a static vector with 9 elements including age, gender, etc. Each patient is associated with a binary label indicating length of stay in ICU, where negative label means hospitalization is not longer than 3 days and positive label marks hospitalization is longer than 3 days. P12 is imbalanced with $\sim$93% positive samples.

**PAM: PAMAP2 Physical Activity Monitoring.** PAM dataset measures daily living activities of 9 subjects with 3 inertial measurement units. We modify it to suit our scenario of irregular time series classification. We excluded the ninth subject due to short length of sensor readouts. We segment the continuous signals into samples with the time window of 600 and the overlapping rate of 50%. PAM originally has 18 activities of daily life. We exclude the ones associated with less than 500 samples, remaining 8 activities. PAM dataset (Reiss & Stricker, 2012) contains 5,333 segments (samples) of sensory signals. Each sample is measured by 17 sensors and contains 600 continuous observations with the sampling frequency 100 Hz. To make time series irregular, we randomly remove 60% of observations. To keep fair comparison, the removed observations are randomly selected but kept the same for all experimental settings and approaches. PAM is labelled by 8 classes where each class represents an activity of daily living. PAM does not include static attributes and the samples are approximately balanced across all 8 categories.

To feed given data into neural networks, we set the input as zero if no value was measured. In highly imbalanced datasets (P19 and P12) we perform batch minority class upsampling, which means that every processed batch has the same number of positive and negative class samples.

## A.3 PERFORMANCE METRICS

Since P19 and P12 datasets are imbalanced, we use the Area Under a ROC Curve (AUROC) and Area Under Precision-Recall Curve (AUPRC) to measure performance. As the PAM dataset is nearly balanced, we also report accuracy, precision, recall and F1 score. We report mean and standard deviation values over 5 independent runs. Model parameters that achieve the best AUROC value on the validation set are used for test set.

### A.4 FURTHER DETAILS ON MODEL HYPERPARAMETERS

**Baseline hyperparameters.** The implementation of baselines follows the corresponding papers including SeFT (Horn et al., 2020), GRU-D (Che et al., 2018), and mTAND (Shukla & Marlin, 2021). We follow the settings of Transformer baseline in (Horn et al., 2020) while implementing Transformer in our work. For average imputation in Trans-mean, we replace the missing values by the global mean value of observations in the sensor (Shukla & Marlin, 2020). We use batch size of 128 and learning rate of 0.0001. Note that we upsample the minority class in each batch to make the batch balance (64 positive samples and 64 negative samples in each batch).

The chosen hyperparameters are the same across datasets (P19, P12, PAM), models (both baselines and RAINDROP), and experimental settings. Remarkably, we found that all the baselines make dummy predictions (classify all testing samples as the majority label) on PAM in Setting 2-3 while RAINDROP makes reasonable predictions. For the comparison to make sense, we use learning rate of 0.001 for baselines on PAM. GRU-D has 49 layers while other models have 2 layers. We run all models for 20 epochs, store the parameters that obtain the highest AUROC in the validation set, and use it to make predictions for testing samples. We use the Adam algorithm for gradient-based optimization (Kingma & Ba, 2014).

**RAINDROP hyperparameters.** Next, we report the setting of unique hyperparameters in our RAINDROP. In the generation of observation embedding, we set $\boldsymbol{R}_u$ as a 4-dimensional vector, thus the produced observation embedding has 4 dimensions. The dimensions of time representation $\boldsymbol{p}^t$ and $\boldsymbol{r}_v$ are both 16. The trainable weight matrix $\boldsymbol{D}$ has shape of $4 \times 32$. The dimensions of $\boldsymbol{w}_u$ and $\boldsymbol{w}_v$ are the same as the number of sensors: 34 in P19, 36 in P12, and 17 in PAM. We set the number of RAINDROP layers $L$ as 2 while the first layer prunes edges and the second layer does not. We set the proportion of edge pruning as 50% (K=50), which means we remove half of the existing edges that have the lowest weights. The $d_k$ is set to 20, while the shape of $\boldsymbol{W}$ is $20 \times 20$. All the activation functions, without specific clarification, are sigmoid functions. The $d_a$ is set equal to the number of sensors. The first layer of $\varphi$ has 128 neurons while the second layer has $C$ neurons (*i.e.*, 2 for P19 and P12; 8 for PAM). We set $\lambda = 0.02$ to adjust the scale of regularization term $\mathcal{L}_r$. All the preprocessed datasets and implementation codes are made available online. More details can be found through our publicly accessible link.

### A.5 FURTHER DETAILS ON SETUP DETAILS FOR SETTING 2

In Setting 2, the selected missing sensors are fixed across different models and chosen in the following way. First, we calculate the importance score for each sensor and rank them in a descending order. The importance score is based on information gain, which we calculate with feeding the observations into a Random Forest classifier with 20 decision trees. Next, we treat each sample as only having one sensor, then feed the single sensor into random forest classifier and record the AUROC. When we have sensors ranked by their AUROC values, we choose the first $n$ sensors (the ones with highest AUROC values) and replace all observations in these sensors by zeros in all samples in validation and test set. The number of missing sensors $n$ is defined indirectly from the user with the sensors' missing ratio in range $[0, 1]$.

### A.6 RESULTS FOR P19 (SETTINGS 2-3)

Here we report the experimental results for P19 in Setting 2 (Table 4) and Setting 3 (Table 5).

**Table 4:** Classification on samples with fixed missing sensors (P19; Setting 2)

| Models | Missing ratio | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0% | | 10% | | 20% | | 30% | | 40% | | 50% | |
| | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC |
| Transformer | 83.2 ± 1.3 | 47.6 ± 3.8 | 77.4 ± 3.5 | 38.2 ± 4.2 | 75.7 ± 3.4 | 35.2 ± 5.4 | 75.1 ± 3.5 | 35.5 ± 4.4 | 75.3 ± 3.5 | 36.2 ± 4.2 | 74.9 ± 3.1 | 35.5 ± 5.0 |
| Trans-mean | 84.1 ± 1.7 | 47.4 ± 1.4 | 79.2 ± 2.7 | 40.6 ± 5.7 | 79.8 ± 2.5 | 38.3 ± 2.8 | 76.9 ± 2.4 | 37.5 ± 5.9 | 76.4 ± 2.0 | 36.3 ± 5.8 | 74.1 ± 2.3 | 41.3 ± 4.7 |
| GRU-D | 83.9 ± 1.7 | 46.9 ± 2.1 | 79.6 ± 2.2 | 37.4 ± 2.5 | 77.5 ± 3.1 | 36.5 ± 4.6 | 76.6 ± 2.9 | 35.1 ± 2.4 | 74.6 ± 2.7 | 35.9± 2.7 | 74.1 ± 2.9 | 33.2 ± 3.8 |
| SeFT | 78.7 ± 2.4 | 31.1 ± 2.8 | 77.3 ± 2.4 | 25.5 ± 2.3 | 63.5 ± 2.0 | 14.0 ± 1.1 | 62.3 ± 2.1 | 12.9 ± 1.2 | 57.8 ± 1.7 | 9.8 ± 1.1 | 56.0 ± 3.1 | 7.8 ± 1.3 |
| mTAND | 80.4 ± 1.3 | 32.4 ± 1.8 | 79.7 ± 2.2 | 29.0 ± 4.3 | 77.8 ± 1.9 | 25.3 ± 2.4 | 77.7 ± 1.9 | 27.8 ± 2.6 | 79.4 ± 2.0 | 32.1 ± 2.1 | 77.3 ± 2.1 | 27.0 ± 2.5 |
| RAINDROP | **87.0 ± 2.3** | **51.8 ± 5.5** | **84.3 ± 2.5** | **46.1 ± 3.5** | **81.9 ± 2.1** | **45.2 ± 6.4** | **81.4 ± 2.1** | **43.7 ± 7.2** | **81.8 ± 2.2** | **44.9 ± 6.6** | **79.7 ± 1.9** | **43.8 ± 5.6** |

**Table 5:** Classification on samples with random missing sensors (P19; Setting 3)

| Models | Missing ratio | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0% | | 10% | | 20% | | 30% | | 40% | | 50% | |
| | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC | AUROC | AUPRC |
| Transformer | 83.2 ± 1.3 | 47.6 ± 3.8 | 82.2 ± 2.7 | 46.8 ± 3.5 | 81.6 ± 3.5 | 42.5 ± 8.5 | 81.3 ± 3.1 | 42.1 ± 4.5 | 80.2 ± 2.9 | 41.9 ± 6.8 | 79.2 ± 1.9 | 43.7 ± 3.7 |
| Trans-mean | 84.1 ± 1.7 | 47.4 ± 1.4 | 82.5 ± 3.7 | 44.7 ± 6.8 | 81.7 ± 2.0 | 45.9 ± 3.6 | 81.2 ± 2.2 | 43.2 ± 6.3 | 80.2 ± 1.7 | 41.5 ± 4.8 | 79.8 ± 3.1 | 39.3 ± 5.1 |
| GRU-D | 83.9 ± 1.7 | 46.9 ± 2.1 | 81.2 ± 3.4 | 46.4 ± 2.7 | 78.6 ± 4.1 | 43.3 ± 2.4 | 76.3 ± 2.5 | 28.5 ± 2.1 | 74.2 ± 2.7 | 29.6 ± 3.1 | 74.6 ± 3.5 | 26.5 ± 4.2 |
| SeFT | 78.7 ± 2.4 | 31.1 ± 2.8 | 76.8 ± 2.2 | 28.3 ± 2.5 | 77.0 ± 2.2 | 24.1 ± 2.4 | 75.2 ± 2.2 | 22.5 ± 3.0 | 73.6 ± 2.7 | 18.3 ± 3.2 | 72.6 ± 2.5 | 15.7 ± 1.9 |
| mTAND | 80.4 ± 1.3 | 32.4 ± 1.8 | 75.2 ± 2.5 | 24.5 ± 2.4 | 74.4 ± 3.5 | 24.6 ± 3.5 | 74.2 ± 3.2 | 22.6 ± 2.3 | 74.1 ± 2.6 | 23.1 ± 3.6 | 73.9 ± 3.7 | 24.6 ± 3.7 |
| RAINDROP | **87.0 ± 2.3** | **51.8 ± 5.5** | **85.5 ± 2.1** | **50.2 ± 5.5** | **83.5 ± 3.2** | **47.4 ± 7.0** | **83.1 ± 1.5** | **48.2 ± 4.7** | **82.6 ± 1.7** | **48.0 ± 5.5** | **80.9 ± 2.4** | **45.2 ± 6.9** |

### A.7 FURTHER DETAILS ON ABLATION STUDY

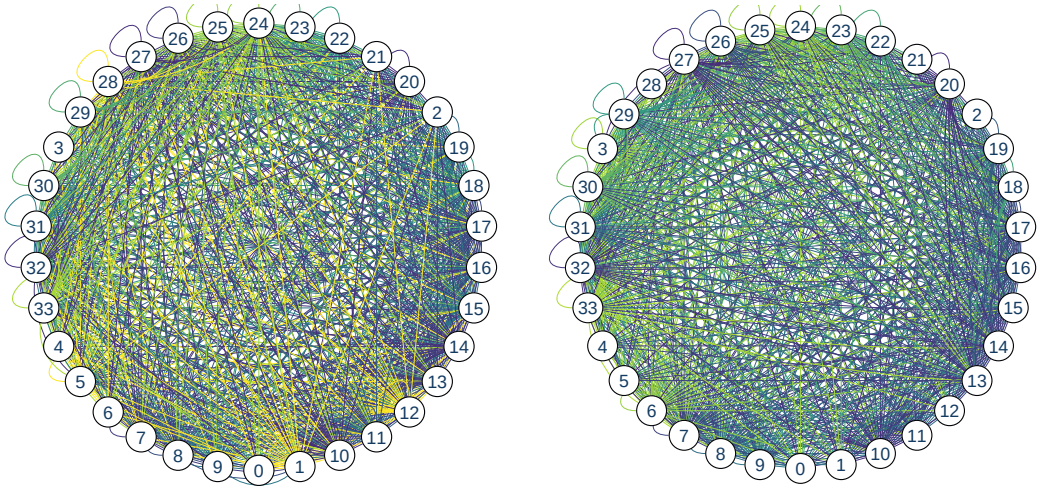
We provide ablation study, taking PAM at Setting 1 as an example, in Table 6. Experimental results show that the full RAINDROP model achieves the best performance, indicating every component or designed structure is useful to the model.

**Table 6:** Results of ablation study on the PAM dataset (Setting 1).

| RAINDROP Model | | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|---|
| W/o weights vector $\boldsymbol{R}_u$ | | 81.1 ± 2.6 | 81.9 ± 2.4 | 80.1 ± 1.6 | 81.6 ± 2.1 |
| W/o inter-sensor dependency | W/o $e_{i,uv}$ | 82.6 ± 1.2 | 82.9± 1.6 | 84.3± 1.4 | 83.8 ± 1.7 |
| | W/o $\boldsymbol{r}_v$ | 86.5 ± 2.4 | 83.3 ± 1.9 | 82.6± 1.5 | 82.9 ± 1.4 |
| | W/o $\boldsymbol{p}_i^t$ | 79.8 ± 2.7 | 80.1 ± 3.6 | 80.6 ± 1.7 | 80.2 ± 2.9 |
| | W/o $\alpha_{i,uv}^t$ | 85.2 ± 2.5 | 86.4 ± 2.7 | 84.5 ± 2.9 | 85.6 ± 2.9 |
| W/o temporal attention | | 81.5 ± 1.9 | 84.6± 1.7 | 83.9 ± 2.5 | 84.2 ± 2.2 |
| W/o sensor level concatenation | | 84.4 ± 2.1 | 86.7± 1.1 | 85.2± 1.9 | 85.8 ± 2.6 |
| W/o regularization term $\mathcal{L}_r$ | | 87.3 ± 2.9 | 88.6± 3.4 | 87.1± 2.8 | 87.6 ± 3.1 |
| Full RAINDROP | | **88.5±1.5** | **89.9±1.5** | **89.9±0.6** | **89.8±1.0** |

### A.8 VISUALIZATION OF INTER-SENSOR DEPENDENCY GRAPHS LEARNED BY RAINDROP

We visualize the learned inter-sensor dependencies ($e_{i,uv}$ before the averaging operation in Eq. 3) on P19 on early sepsis prediction. The data shown are for testing set of P19 including 3,881 samples (3708 negative and 173 positive). As RAINDROP learns the specific graph for each sample, we take average of all positive samples and visualize it in Figure 4(a); and visualize the average of all negative samples in Figure 4(b). As we take average, the edges with weights smaller than 0.1 (means they rarely appear in graphs) are ignored. The averaged edge weights range from 0.1 to 1. We initialize all sample graphs as complete graph that has $1{,}156 = 34 \times 34$ edges, then prune out 50% of them in training phase, remaining 578 edges. The 34 nodes in figures denote 34 sensors measured in P19, as listed https://physionet.org/content/challenge-2019/1.0.0/. We list the sensor names here: 0: HR; 1: O2Sat; 2: Temp; 3: SBP; 4: MAP; 5: DBP; 6: Resp; 7: EtCO2; 8: BaseExcess; 9: HCO3; 10: FiO2; 11: pH; 12: PaCO2; 13: SaO2; 14: AST; 15: BUN; 16: Alkalinephos; 17: Calcium; 18: Chloride; 19: Creatinine; 20: Bilirubin_direct; 21: Glucose; 22: Lactate; 23: Magnesium; 24: Phosphate; 25: Potassium; 26: Bilirubin_total; 27: TroponinI; 28: Hct; 29: Hgb; 30: PTT; 31: WBC; 32: Fibrinogen; 33: Platelets.




**(a)** Negative samples

**(b)** Positive samples

**Figure 4:** Learned structure for negative and positive samples (P19; Setting 1). The nodes numbered from 0 to 33 denote 34 sensors used in P19 (sensor names are listed in Appendix A.8). To make the visualized structures easier to understand, we use darker green to denote higher weight value and while yellow to denote lower weight value. We can observe distinguishable patterns across two learned sensor dependency graphs, indicating RAINDROP is able to adaptively learn graph structures that are sensitive to the classification task. For example, we find that the nodes 1 (pulse oximetry), 4 (mean arterial pressure), 5 (diastolic BP), and 12 (partial pressure of carbon dioxide from arterial blood) have lower weights in negative samples.