GraFITi: Graphs for Forecasting Irregularly Sampled Time Series

Vijaya Krishna Yalavarthi[1], Kiran Madhusudhanan[1], Randolf Scholz[1], Nourhan Ahmed[1], Johannes Burchert[1], Shayan Jawed[1], Stefan Born[2], Lars Schmidt-Thieme[1]

[1] Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany
[2] Institute of Mathematics, TU Berlin, Germany
{yalavarthi, kiranmadhusud, scholz, ahmedn, burchert, shayan, schmidt-thieme}@ismll.uni-hildesheim.de, born@math.tu-berlin.de

Abstract

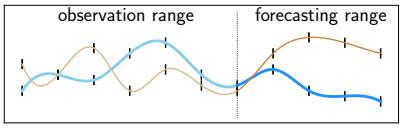
Forecasting irregularly sampled time series with missing values is a crucial task for numerous real-world applications such as healthcare, astronomy, and climate sciences. State-of-the-art approaches to this problem rely on Ordinary Differential Equations (ODEs) which are known to be slow and often require additional features to handle missing values. To address this issue, we propose a novel model using Graphs for Forecasting Irregularly Sampled Time Series with missing values which we call GraFITi. GraFITi first converts the time series to a Sparsity Structure Graph which is a sparse bipartite graph, and then reformulates the forecasting problem as the edge weight prediction task in the graph. It uses the power of Graph Neural Networks to learn the graph and predict the target edge weights. GraFITi has been tested on 3 real-world and 1 synthetic irregularly sampled time series dataset with missing values and compared with various state-of-the-art models. The experimental results demonstrate that GraFITi improves the forecasting accuracy by up to 17% and reduces the run time up to 5 times compared to the state-of-the-art forecasting models.

1 Introduction

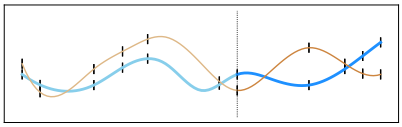
Time series forecasting predicts future values based on past observations. While extensively studied, most research focuses on regularly sampled and fully observed multivariate time series (MTS) (Lim and Zohren 2021; Zeng et al. 2022; De Gooijer and Hyndman 2006). Limited attention is given to irregularly sampled time series with missing values (IMTS) which is commonly seen in many real-world applications. IMTS has independently observed channels at irregular intervals, resulting in sparse data alignment. The focus of this work is on forecasting IMTS. Additionally, there is another type called irregular multivariate time series which is fully observed at irregular time intervals (Figure 1 illustrates the differences) which is not the interest of this paper.

Ordinary Differential Equations (ODE) model continuous time series, and predict system evolution over time based on the rate of change of state variables as shown in Eq. 1.

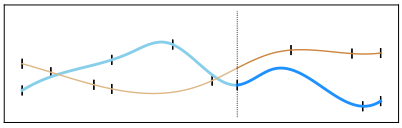
$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = f(t, x(t)) \tag{1}$$



(a) Forecasting regular multivariate time series (MTS)



(b) Forecasting irregular multivariate time series



(c) Forecasting irregularly sampled time series with missing values (IMTS)

Figure 1: Illustrating forecasting task in various multivariate time series.

ODE-based models (Schirmer et al. 2022; De Brouwer et al. 2019; Biloš et al. 2021; Scholz et al. 2023) are able to forecast at arbitrary time points. However, ODE models can be slow because of their auto-regressive nature and computationally expensive numerical integration process. Also, some ODE models cannot directly handle missing values in the observation space, hence, they often rely on missing value indicators (De Brouwer et al. 2019; Biloš et al. 2021) which are given as additional channels in the data.

To circumvent the above challenges, in this work, we propose a novel model called GraFITi: graphs for forecasting IMTS. GraFITi converts IMTS data into a Sparsity Structure Graph and formulates forecasting as edge weight prediction in the graph. This approach represents channels and timepoints as disjoint nodes connected by observation values as edges in a bipartite graph. GraFITi uses multiple graph neural network (GNN) layers with attention and feed-forward mechanisms to learn node and edge interactions. Our Spar-

sity Structure Graph, by design, provides a more dynamic and adaptive approach to process IMTS data, and improves the performance of the forecasting task.

We evaluated GraFITi for forecasting IMTS using 3 real-world and 1 synthetic dataset. Comparing it with state-of-the-art methods for IMTS and selected baselines for MTS, we show that GraFITi provides superior forecasts. Our contributions are summarized as follows:

- We introduce a novel representation of irregularly sampled time series with missing values (IMTS) as sparse bipartite graphs, the Sparsity Structure Graph, that efficiently can handle missing values in the observation space of the time series (Section 4).
- We propose a novel model based on this representation, GraFITi, that can leverage any graph neural network to perform time series forecasting for IMTS (section 5).
- We provide extensive experimental evaluation on 3 real world and 1 synthetic dataset that shows that GraFITi improves the forecasting accuracy of the best existing models by up to 17% and the run time improvement up to 5 times (section 6). Implementation code: https://github.com/yalavarthivk/GraFITi.

2 Related Work

This work focuses on the forecasting of irregularly sampled time series data with missing values using graphs. We discuss the research done in: IMTS forecasting models, Graphs for MTS, and models for edge weight prediction in graphs.

Forecasting of IMTS Research on IMTS has mainly focused on classification (Yalavarthi, Burchert, and Schmidt-Thieme 2022; Li and Marlin 2015; Lipton, Kale, and Wetzel 2016; Rubanova, Chen, and Duvenaud 2019; Shukla and Marlin 2021; Horn et al. 2020; Tashiro et al. 2021) and interpolation (Che et al. 2018; Rubanova, Chen, and Duvenaud 2019; Shukla and Marlin 2021; Tashiro et al. 2021; Shukla and Marlin 2022; Yalavarthi, Burchert, and Schmidt-Thieme 2023), with limited attention to forecasting tasks. Existing models for these tasks mostly rely on Neural ODEs (Che et al. 2018). In Latent-ODE (Rubanova, Chen, and Duvenaud 2019), an ODE was combined with a Recurrent Neural Network (RNN) for updating the state at the point of new observation. The GRU-ODE-Bayes model (De Brouwer et al. 2019) improved upon this approach by incorporating GRUs, ODEs, and Bayesian inference for parameter estimation. The Continuous Recurrent Unit (CRU) (Schirmer et al. 2022) based model uses a state-space model with stochastic differential equations and kalman filtering. The recent LinODENet model (Scholz et al. 2023) enhanced CRU by using linear ODEs and ensures self-consistency in the forecasts. Another branch of study involves Neural Flows (Biloš et al. 2021), which use neural networks to model ODE solution curves, rendering the ODE integrator unnecessary. Among various flow architectures, GRU flows have shown good performance.

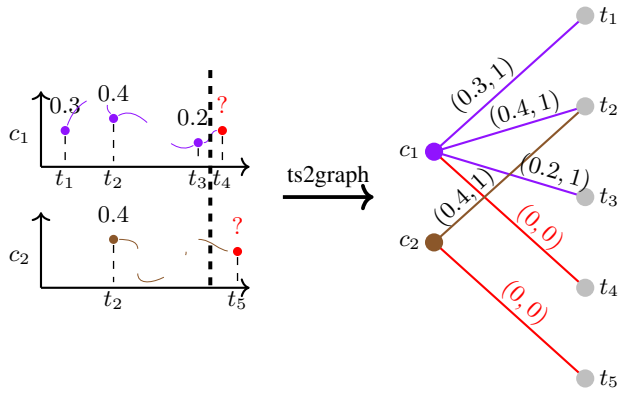
Using graphs for MTS In addition to CNNs, RNNs, and Transformers, graph-based methods have been studied for MTS tasks such as forecasting, imputation and reconstruction. Early GNN-based approaches, such as (Wu et al. 2020), required a pre-defined adjacency matrix to establish relationships between the time series channels. More recent models like the Spectral Temporal Graph Neural Network (Cao et al. 2020) and the Time-Aware Zigzag Network (Chen, Segovia, and Gel 2021) improved on this by using GNNs to capture dependencies between variables in the time series. On the other hand, Satorras, Rangapuram, and Januschowski (2022) proposed a bipartite setup with induced nodes to reduce graph complexity, built solely from the channels. Existing graph-based time series forecasting models focus on learning correlations or similarities between channels, without fully exploiting the graph structure. Note that although graphs have been used for MTS forecasting, state-of-the-art approaches apply attention (Zhou et al. 2021, 2022) or linear layers (Zeng et al. 2022). It is not trivial to directly apply them to IMTS due to presence of missing values and irregular sampling. Recently, GNNs were used for imputation and reconstruction of MTS with missing values, treating MTS as sequences of graphs where nodes represent sensors and edges denote correlation (Cini, Marisca, and Alippi 2022; Marisca, Cini, and Alippi 2022; Ahmed and Schmidt-Thieme 2023). Similar to previous studies, they learn similarity or correlation among channels.

Graph Neural Networks for edge weight prediction Graph Neural Networks (GNNs) are designed to process graph-based data. While most GNN literature such as Graph Convolutional Networks, Graph Attention Networks focuses on node classification (Kipf and Welling 2017; Velickovic et al. 2017), a few studies have addressed edge weight prediction. Existing methods (De Sá and Prudêncio 2011; Fu et al. 2018) in this domain rely on latent features and graph heuristics, such as node similarity (Zhao et al. 2015), proximity measures (Murata and Moriyasu 2007), and local rankings (Yang and Wang 2020). Recently, deep learning-based approaches (Hou and Holder 2017; Zulaika et al. 2022; You et al. 2020) were proposed. Another branch of research deals with edge weight prediction in weighted signed graphs (Kumar et al. 2016) tailored to social networks. However, all proposed methods typically operate in a transductive setup with a single graph split into training and testing data, which may not be suitable for cases involving multiple graphs like ours, where training and evaluation need to be done on separate graph partitions.

3 The Time Series Forecasting Problem

An **irregularly sampled times series with missing values**, is a finite sequence of pairs $S = (t_n, x_n)_{n=1:N}$ where $t_n \in \mathbb{R}$ is the n-th **observation timepoint** and $x_n \in (\mathbb{R} \cup \{\texttt{NaN}\})^C$ is the n-th **observation event**. Components with $x_{n,c} \neq \texttt{NaN}$ represent **observed values** by channel c at event time t_n, and $x_{n,c} = \texttt{NaN}$ represents a missing value. C is the total number of channels.

A **time series query** is a pair (Q, S) of a time series S and a sequence $Q = (q_k, c_k)_{k=1:K}$ such that the value of channel $c_k \in \{1, \ldots, C\}$ is to be predicted at time $q_k \in \mathbb{R}$. We call a query a **forecasting query**, if all its query timepoints are



(a) IMTS (b) Sparsity Structure Graph

Figure 2: Representation of IMTS (a) as Sparsity Structure Graph (b). Timpoints and channels are the nodes, and observation measurements are the edges. Target edges are differentiated from observed ones with target indicator (2^{nd} value in the edge feature).

after the last timepoint of the time series S, an **imputation query** if all of them are before the last timepoint of S and a **mixed query** otherwise. In this paper, we are interested in forecasting only.

A vector $y \in \mathbb{R}^K$ we call an **answer** to the forecasting query: y_k is understood as the predicated value of time series S at time q_k in channel c_k. The difference between two answers y, y' to the same query can be measured by any loss function, for example by a simple squared error

$$\ell(y, y') \coloneqq \frac{1}{K} \sum_{k=1}^{K} (y_k - y'_k)^2$$

The **time series forecasting problem** is as follows: given a dataset of pairs $D \coloneqq (Q_i, S_i, y_i)_{i=1:M}$ of forecasting queries and ground truth answers from an unknown distribution p^{data} and a loss function ℓ on forecasting answers, find a forecasting model $\hat{y}$ that maps queries (Q, S) to answers $\hat{y}(Q, S)$ such that the expected loss between ground truth answers and forecasted answers is minimal:

$$\mathcal{L}(\hat{y}; p^{\text{data}}) \coloneqq \mathbb{E}_{(Q,S,y)\sim p^{\text{data}}} \left[\ell(y, \hat{y}(Q, S)) \right]$$

4 Sparsity Structure Graph Representation

We describe the proposed Sparsity Structure Graph representation and convert the forecasting problem as an edge weight prediction problem. Using this representation:

- We explicitly obtain the relationship between the channels and timepoints via observation values allowing the inductive bias of the data to pass into the model.
- We elegantly handle the missing values in IMTS in the observation space by connecting edges only for the observed values.

Missing values represented by `NaN`-values are unsuited for standard arithmetical operations. Therefore, they are often encoded by dedicated binary variables called **missing value indicators** or masks: $x_n \in (\mathbb{R} \times \{0, 1\})^C$. Here, $(x_{n,c,1}, 1)$ encodes an observed value and $(0, 0)$ encodes a missing value. Usually, both components are seen as different scalar variables: the real value $x_{n,c,1}$ and its binary missing value indicator / mask $x_{n,c,2}$, the relation between both is dropped and observations simply modeled as $x_n \in \mathbb{R}^{2C}$. The main motivation behind the proposed Sparsity Structure Graph is to represent IMTS avoiding missing value indicators in the observation space.

We propose a novel representation of a time series S using a bipartite graph $G = (V, E)$. The graph has nodes for channels and timepoints, denoted as V_C and V_T respectively ($V = V_C \dot\cup V_T$). Edges $E \subseteq V_C \times V_T$ connect each channel node to its corresponding timepoint node with an observation. Edge features F^{edge} are the observation values and node features F^{node} are the channel IDs and unique timepoints. Nodes $V_C \coloneqq \{1, \ldots, C\}$ represent channels and nodes $V_T \coloneqq \{C+1, \ldots, C+N\}$ represent timepoints:

$$\begin{aligned}
V &\coloneqq \{1, \ldots, C+N\} = V_C \dot\cup V_T \\
E &\coloneqq \{\{i, j\} \mid x_{i-C,j} \neq \texttt{NaN}, i \in V_T, j \in V_C\} \\
F_v^{\text{node}} &\coloneqq \begin{cases} v : & v \in V_C \\ t_j : & v \in V_T,\ j = v - C \end{cases} \\
F_e^{\text{edge}} &\coloneqq x_{i-C,j} \quad \text{for} \quad e = \{i, j\} \in E \quad \text{with} \quad i \in V_T, j \in V_C
\end{aligned} \tag{2}$$

For an IMTS, missing values make the bipartite graph sparse, meaning $|E| \ll C \cdot N$. However, for a fully observed time series, where there are no missing values, i.e. $|E| = C \cdot N$, the graph is a complete bipartite graph.

We extend this representation to time series queries (S, Q) by adding additional edges between queried channels and timepoints, and distinguish observed and queried edges by an additional binary edge feature called target indicator. *Note that the target indicator used to differentiate the observed edge and target edge is different from the missing value indicator which is used to represent the missing observations in the observation space.* Given a query $Q = (q_k, c_k)_{k=1:K}$, let $(t'_1, \ldots, t'_{K'})$ be an enumeration of the unique queried timepoints q_k. We introduce additional nodes $V_Q \coloneqq \{C+N+1, \ldots, C+N+K'\}$ so that the augmented graph, together with the node and edge features is given as

$$\begin{aligned}
V &\coloneqq V_C \dot\cup V_T \dot\cup V_Q = \{1, \ldots, C+N+K'\} \\
E &\coloneqq \{\{i, j\} \mid x_{i-C,j} \neq \texttt{NaN}, i \in V_T, j \in V_C\} \\
&\quad \cup \{\{i, j\} \mid i \in V_Q, j \in V_C, (t'_{i-N-C}, j) \in Q\} \\
F_v^{\text{node}} &\coloneqq \begin{cases} v : & v \in V_C \\ t_j : & v \in V_T, j = v - C \\ t'_j : & v \in V_Q, j = v - C - N \end{cases} \\
F_e^{\text{edge}} &\coloneqq \begin{cases} (x_{i,j,1}, 1) : & e = \{i, j\} \in E, \quad i \in V_T, j \in V_C \\ (0, 0) \quad\ : & e = \{i, j\} \in E, \quad i \in V_Q, j \in V_C \end{cases}
\end{aligned} \tag{3}$$

where $(t'_{i-N-C}, j) \in Q$ is supposed to mean that

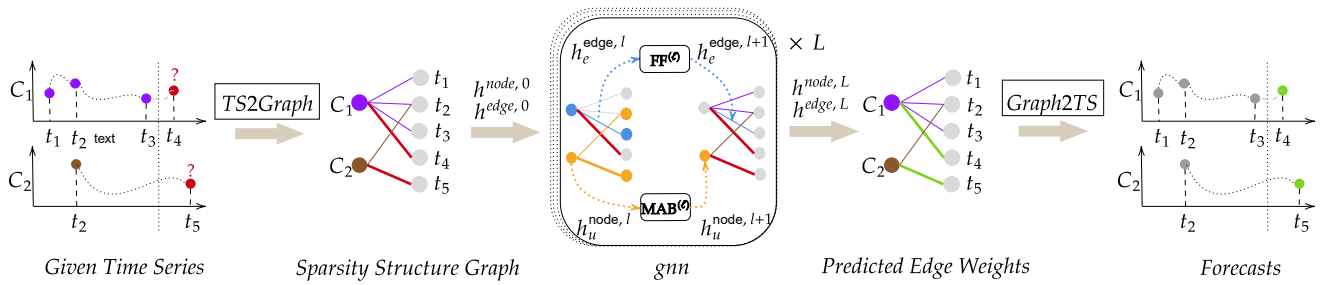


Figure 3: Overall architecture of GraFITi. First a Sparsity Structure Graph is created from the given IMTS using TS2Graph. Then, we update edge and node embeddings using L many gnn layers. Final predictions are made using Graph2TS.

(t'_{i-N-C}, j) appears in the sequence Q. To denote this graph representation, we write briefly

$$\text{ts2graph}(X, Q) := (V, E, F^{\text{node}}, F^{\text{edge}}) \tag{4}$$

The conversion of an IMTS to a Sparsity Structure Graph is shown in Figure 2.

To make the graph representation $(V, E, F^{\text{node}}, F^{\text{edge}})$ of a time series query processable by a graph neural network, node and edge features have to be properly embedded, otherwise, both, the nominal channel ID and the timepoint are hard to compute on. We propose an **Initial Embedding layer** that encodes channel IDs via a onehot encoding and time points via a learned sinusoidal encoding (Shukla and Marlin 2021):

$$h_v^{\text{node},0} := \begin{cases} \mathbf{FF}(\text{onehot}(F_v^{\text{node}})) & : v \in V_C \\ \sin(\mathbf{FF}(F_v^{\text{node}})) & : v \in V_T \dot\cup V_Q \end{cases} \tag{5}$$

$$h_e^{\text{edge},0} := \mathbf{FF}(F_e^{\text{edge}}) \quad \text{for} \quad e \in E \tag{6}$$

where onehot denotes the binary indicator vector and **FF** denotes a separate fully connected layer in each case.

The final graph neural network layer $(h^{\text{node},L}, h^{\text{edge},L})$ has an embedding dimension 1. The scalar values of the query edges are taken as the predicted answers to the encoded forecasting query:

$$\hat{y} := \text{graph2ts}(h^{\text{node},L}, h^{\text{edge},L}, V, E) = (h_{e_k}^{\text{edge},L})_{k=1:K}$$
$$\text{where } e_k = \{C + N + k', c_k\} \quad \text{with} \quad t'_{k'} = q_k \tag{7}$$

5 Forecasting with GraFITi

GraFITi first encodes the time series query to graph using Eq. 4 and compute initial embeddings for the nodes $(h^{\text{node},0})$ and edges $(h^{\text{edge},0})$ using Eqs. 5 and 6 respectively. Now, we can leverage the power of graph neural networks for further processing the encoded graph. Node and edge features are updated layer wise, from layer l to $l+1$ using a graph neural network:

$$(h^{\text{node},l+1}, h^{\text{edge},l+1}) := \text{gnn}^{(l)}(h^{\text{node},l}, h^{\text{edge},l}, V, E) \tag{8}$$

There have been a variety of gnn architectures such as Graph Convolutional Networks (Kipf and Welling 2017), Graph Attention Networks (Velickovic et al. 2017), proposed in the literature. In this work, we propose a model adapting the Graph Attention Network (Velickovic et al. 2017) to our graph setting and incorporate essential components for handling sparsity structure graphs. While a Graph Attention Network computes attention weights by adding queries and keys, we found no advantage in using this approach. Thus, we utilize standard attention mechanism, in our attention block. Additionally, we also use edge embeddings in our setup to update node embeddings in a principled manner.

5.1 Graph Neural Network (gnn)

First, we define Multi-head Attention block (**MAB**) and Neighborhood functions that are used in our gnn.

A Multi-head attention block (**MAB**) (Vaswani et al. 2017) is represented as:

$$\mathbf{MAB}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) := \alpha(\mathcal{H} + \mathbf{FF}(\mathcal{H}))$$
$$\text{where} \quad \mathcal{H} := \alpha(\mathcal{Q} + \mathbf{MHA}(\mathcal{Q}, \mathcal{K}, \mathcal{V})) \tag{9}$$

where $\mathcal{Q}, \mathcal{K}$ and, $\mathcal{V}$ are called queries, keys, and values respectively, **MHA** is multi-head attention (Vaswani et al. 2017), α is a non-linear activation.

The Neighborhood of a node u is defined as the set of all the nodes connected to u through edges in E:

$$\mathcal{N}(u) := \{v \mid \{u, v\} \in E\} \tag{10}$$

GraFITi consists of L gnn layers. In each layer, node embeddings are updated using neighbor node embeddings and edge embeddings connecting them. For edge embeddings, we use the embeddings of the adjacent nodes and the current edge embedding. The overall architecture of GraFITi is shown in Figure 3.

Update node embeddings To update embedding of a node $u \in V$, first, we create a sequence of features H_u concatenating its neighbor node embedding $h_v^{\text{node},l}$ and edge embedding $h_e^{\text{edge},l}, e = \{u, v\}$ where $v \in \mathcal{N}(u)$. We then pass $h_u^{\text{node},l}$ as queries and H_u as keys and values to $\mathbf{MAB}$.

$$h_u^{\text{node},l+1} := \mathbf{MAB}^{(l)}\left(h_u^{\text{node},l}, H_u, H_u\right) \tag{11}$$

$$H_u := \left([h_v^{\text{node},l} \parallel h_e^{\text{edge},l}]\right)_{v \in \mathcal{N}(u)}, \; e = \{u, v\} \tag{12}$$

Updating edge embeddings: To compute edge embedding $h_e^{\text{edge},l+1}, e = \{u, v\}$ we concatenate $h_u^{\text{node},l}, h_v^{\text{node},l}$ and

Algorithm 1: Graph Neural Network ($\text{gnn}^{(l)}$)

Require: $h^{\text{node},l}, h^{\text{edge},l}, V, E$
  for $u \in V$ **do**
    $H_u \leftarrow \left([h_v^{\text{node},l} \parallel h_e^{\text{edge},l}]\right)_{v \in \mathcal{N}(u)}$ //$e = \{u, v\}$
    $h_u^{\text{node},l+1} \leftarrow \mathbf{MAB}^{(l)}(h_u^{\text{node},l}, H_u, H_u)$
  for $e = \{u, v\} \in E$ **do**
    $h_e^{\text{edge},l+1} \leftarrow \alpha\left(h_e^{\text{edge},l} + \mathbf{FF}^{(l)}\left([h_u^{\text{node},l} \parallel h_v^{\text{node},l} \parallel h_e^{\text{edge},l}]\right)\right)$
  return $h^{\text{node},l+1}, h^{\text{edge},l+1}$

Algorithm 2: Forward pass of GraFITi

Require: Observed time series forecasting query (S, Q)
  $(V, E, F_{\text{node}}, F_{\text{edge}}) \leftarrow \text{ts2graph}(S, Q)$ //Eq. 4
  //Initial embeddings of nodes and edges
  $h^{\text{node},0} \leftarrow \{h_u^{\text{node},0} \mid u \in V\}$ //Eq. 5
  $h^{\text{edge},0} \leftarrow \{h_{u,v}^{\text{edge},0} \mid \{u, v\} \in E\}$ //Eq: 6
  //Graph Neural Network
  for $l \in \{0, \ldots, L-1\}$ **do**
    $h^{\text{node},l+1}, h^{\text{edge},l+1} \leftarrow \text{gnn}^{(l)}(h^{\text{node},l}, h^{\text{edge},l}, V, E)$ //Alg. 1
  $\hat{y} \leftarrow \text{graph2ts}(h^{\text{node},L}, h^{\text{edge},L}, V, E)$ //Eq: 7
  return $\hat{y}$

$h_e^{\text{edge},l}$, and pass it through a dense layer ($\mathbf{FF}$) followed by a residual connection and nonlinear activation.

$$h_e^{\text{edge},l+1} \coloneqq \alpha\left(h_e^{\text{edge},l} + \mathbf{FF}^{(l)}\left(h_u^{\text{node},l} \parallel h_v^{\text{node},l} \parallel h_e^{\text{edge},l}\right)\right) \tag{13}$$

where $e = \{u, v\}$. *Note that, although edges are undirected, edge embedding is computed by concatenating the embeddings in a specific order i.e., the channel embedding, time embedding and edge embedding.* We show the process of updating nodes and edges in layer l using a gnnin Algorithm 1.

Answering the queries As mentioned in Section 4, our last $\text{gnn}^{(L)}$ layer has embedding dimension 1. Hence, after processing the graph features through L many gnn layers, we use Eq. 7 to decode the graph and provide the predicted answers to the time series query. A forward pass of GraFITi is presented in Algorithm 2.

Computational Complexity The computational complexity of GraFITi primarily comes from using MAB in Eq. 11. For a single channel node u, the maximum complexity for computing its embedding is $\mathcal{N}(u)$ since only neighborhood connections are used for the update, and $\mathcal{N}(u) \subseteq \{C+1, ..., C+N+K'\}$. Thus, computing the embeddings of all channel nodes is $\mathcal{O}(|E|)$. Similarly, the computational complexity of MAB for computing the embeddings of all nodes in $V_T \dot{\cup} V_Q$ is also $\mathcal{O}(|E|)$.

Delineating from GRAPE (You et al. 2020) You et al. (2020) introduced GRAPE, a graph-based model for imputing and classifying vector datasets with missing values. This approach employs a bipartite graph, with nodes divided into separate sets for features and sample IDs. The edges of this graph represent the feature values associated with the samples. Notably, GRAPE learns in a transductive manner, encompassing all the data samples, including those from the test set, within in the graph. In contrast, GraFITi uses an inductive approach. Here, each instance is a Sparsity Structure Graph, tailored for time series data. In this structure, nodes are divided into distinct sets for channels and timepoints, while the edges are the time series observations.

Name	#Sample	#Chan	Max. len.	Max. Obs.	Sparse
USHCN	1,100	5	290	320	77.9%
MIMIC-III	21,000	96	96	710	94.2%
MIMIC-IV	18,000	102	710	1340	97.8%
Physionet'12	12,000	37	48	520	85.7%

Table 1: Statistics of the datasets used in the experiments. Sparsity means the % missing observations in the time series

6 Experiments

6.1 Dataset Description

4 datasets including 3 medical and 1 synthetic climate IMTS datasets are used for evaluating the proposed model. Basic statistics of the datasets are provided in Table 1.

Physionet'12 (Silva et al. 2012) consists of ICU patient records observed for 48 hours. MIMIC-III (Johnson et al. 2016) is also a medical dataset that contains measurements of the ICU patients observed for 48 hours. MIMIC-IV (Johnson et al. 2021) is built upon the MIMIC-III database. USHCN (Menne, Williams Jr, and Vose 2015) is a climate dataset that consists of the measurements of daily temperatures, precipitation and snow observed over 150 years from 1218 meteorological stations in the USA. For MIMIC-III, MIMIC-IV and USHCN, we followed the pre-processing steps provided by Scholz et al. (2023); Biloš et al. (2021); De Brouwer et al. (2019). Hence, observations in MIMIC-III and MIMIC-IV are rounded for 30 mins and 1 min respectively. Whereas for the Physionet'12, we follow the protocol of Che et al. (2018); Cao et al. (2018); Tashiro et al. (2021) and processed the dataset to have hourly observations.

6.2 Competing Algorithms

We select 4 IMTS forecasting models for comparison, including GRU-ODE-Bayes (De Brouwer et al. 2019), Neural Flows (Biloš et al. 2021), CRU (Schirmer et al. 2022) and LinODENet (Scholz et al. 2023). Additionally, we use the well established IMTS interpolation model mTAN (Shukla and Marlin 2021). It is interesting to verify the performance of well established MTS forecasting models for IMTS setup. We do this by adding missing value indicators as separate channels to the series and process the time series along with the missing value indicators. Hence we compare with the Informer+, Fedformer+, DLinear+ and NLinear+ which are variants of Informer (Zhou et al. 2021), FedFormer (Zhou et al. 2022), DLinear and NLinear (Zeng et al. 2022) respectively. We also compare with the published results from (De Brouwer et al. 2019) for the NeuralODE-VAE (Chen et al. 2018), Sequential VAE (Krishnan, Shalit, and Sontag 2015, 2017), GRU-Simple (Che et al. 2018), GRU-D (Che et al. 2018) and T-LSTM (Baytas et al. 2017).

	USHCN	MIMIC-III	MIMIC-IV	Physionet'12
DLinear+	0.347 ± 0.065	0.691 ± 0.016	0.577 ± 0.001	0.380 ± 0.001
NLinear+	0.452 ± 0.101	0.726 ± 0.019	0.620 ± 0.002	0.382 ± 0.001
Informer+	0.320 ± 0.047	0.512 ± 0.064	0.420 ± 0.007	0.347 ± 0.001
FedFormer+	2.990 ± 0.476	1.100 ± 0.059	2.135 ± 0.304	0.455 ± 0.004
NeuralODE-VAE	(0.960 ± 0.110)	(0.890 ± 0.010)	–	–
Sequential VAE	(0.830 ± 0.070)	(0.920 ± 0.090)	–	–
GRU-Simple	(0.750 ± 0.120)	(0.820 ± 0.050)	–	–
GRU-D	(0.530 ± 0.060)	(0.790 ± 0.060)	–	–
T-LSTM	(0.590 ± 0.110)	(0.620 ± 0.050)	–	–
mTAN	0.300 ± 0.038	0.540 ± 0.036	ME	0.315 ± 0.002
GRU-ODE-Bayes	0.401 ± 0.089	0.476 ± 0.043	0.360 ± 0.001	0.329 ± 0.004
	(0.430 ± 0.070)	(0.480 ± 0.010)	(0.379 ± 0.005)	–
Neural Flow	0.414 ± 0.102	0.477 ± 0.041	0.354 ± 0.001	0.326 ± 0.004
	–	(0.490 ± 0.004)	(0.364 ± 0.008)	–
CRU	0.290 ± 0.060	0.592 ± 0.049	ME	0.379 ± 0.003
LinODEnet	0.300 ± 0.060	0.446 ± 0.033	0.272 ± 0.002	0.299 ± 0.001
	(0.290 ± 0.060)	(0.450 ± 0.020)	(0.274 ± 0.002)	–
GraFITi (ours)	**0.272** ± **0.047** ↑ 9.3%	**0.396** ± **0.030** ↑ 11.2%	**0.225** ± **0.001** ↑ 17.2%	**0.286** ± **0.001** ↑ 4.3%

Table 2: Forecasting next three time steps. Evaluation metric MSE, Lower is better. Best results are in bold. Published results are in open brackets, Physionet'12 dataset was not used by the baseline models hence do not have published results. ↑ indicates % improvement by GraFITi. 'ME' indicates Memory Error.

6.3 Experimental Setup

Task protocol We followed Scholz et al. (2023); Biloš et al. (2021); De Brouwer et al. (2019), applied 5-fold cross-validation and selected hyperparameters using a holdout validation set (20%). For evaluation, we used 10% unseen data. All models were trained on Mean Squared Error, which is also the evaluation metric.

Hyperparamter search We searched the following hyperparameters for GraFITi: $L \in \{1, 2, 3, 4\}$, #heads in **MAB** from $\{1, 2, 4\}$, and hidden nodes in dense layers from $\{16, 32, 64, 128, 256\}$. We randomly sampled sets of 5 different hyperparameters and choose the one that has the best performance on validation dataset. We used the Adam optimizer with learning rate of 0.001, halving it when validation loss did not improve for 10 epochs. All models were trained for up to 200 epochs, using early stopping with a patience to 30 epochs. All the models were experimented using the PyTorch library on a GeForce RTX-3090 GPU.

6.4 Experimental Results

First, we set the observation and prediction range of the IMTS following (Scholz et al. 2023; Biloš et al. 2021; De Brouwer et al. 2019). For the USHCN dataset, the model observes for the first 3 years and forecasts the next 3 time steps. For the medical datasets, the model observes for the first 36 hours in the series and predicts the next 3 time steps. The results, including the mean and standard deviation, are presented in Table 2. The best result is highlighted in bold and the next best in italics. Additionally, we also provide the published results from (Scholz et al. 2023; Biloš et al. 2021; De Brouwer et al. 2019) in brackets for comparison.

The GraFITi model is shown to be superior compared to all baseline models across all the datasets. Specifically, in the MIMIC-III and MIMIC-IV datasets, *GraFITi provides around* 11.2% *and* 17.2% *improvement in forecasting accuracy compared to the next best model LinODEnet*. The results on the USHCN dataset have high variance making it challenging to compare the models on this dataset. However, we experimented on it for completeness. Again, we achieve the best result with 9.2% improvement compared to the next best model. We note that, the MTS forecasting models that are adapted for the IMTS task, perform worse than any of the IMTS forecasting models demonstrating the limitation of MTS models applied to IMTS tasks.

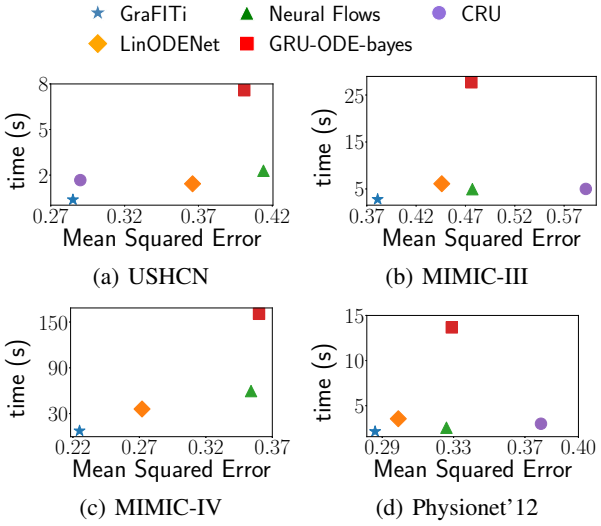


Figure 4: Efficiency comparison: GraFITi, LinODEnet, CRU, Neural Flows and GRU-ODE-Bayes.

Efficiency comparison We compare the efficiency of leading IMTS forecasting models: GraFITi, LinODEnet,

	Obs. / Pred.	GraFITi (ours)	LinODEnet	CRU	Neural Flow	GRU-ODE-Bayes	↑ %
MIMIC-III	24/12	**0.438 ± 0.009**	0.477 ± 0.021	0.575 ± 0.020	0.588 ± 0.014	0.591 ± 0.018	**8.2%**
	24/24	**0.491 ± 0.014**	0.531 ± 0.022	0.619 ± 0.028	0.651 ± 0.017	0.653 ± 0.023	**7.5%**
	36/6	**0.457 ± 0.050**	0.492 ± 0.019	0.647 ± 0.051	0.573 ± 0.043	0.580 ± 0.049	**7.1%**
	36/12	**0.490 ± 0.027**	0.554 ± 0.042	0.680 ± 0.043	0.620 ± 0.035	0.632 ± 0.044	**10.8%**
MIMIC-IV	24/12	**0.285 ± 0.001**	0.335 ± 0.002	ME	0.465 ± 0.003	0.366 ± 0.154	**14.9%**
	24/24	**0.285 ± 0.002**	0.336 ± 0.002	ME	0.465 ± 0.003	0.439 ± 0.003	**15.1%**
	36/6	**0.260 ± 0.002**	0.309 ± 0.002	ME	0.405 ± 0.001	0.393 ± 0.002	**15.9%**
	36/12	**0.261 ± 0.005**	0.309 ± 0.002	ME	0.395 ± 0.001	0.393 ± 0.002	**15.5%**
Physionet'12	24/12	**0.365 ± 0.001**	0.373 ± 0.001	0.435 ± 0.001	0.431 ± 0.001	0.432 ± 0.003	**2.1%**
	24/24	**0.401 ± 0.001**	0.411 ± 0.001	0.467 ± 0.002	0.506 ± 0.002	0.505 ± 0.001	**2.4%**
	36/6	**0.319 ± 0.001**	0.329 ± 0.001	0.396 ± 0.003	0.365 ± 0.001	0.363 ± 0.004	**3.0%**
	36/12	**0.347 ± 0.001**	0.357 ± 0.001	0.417 ± 0.001	0.398 ± 0.001	0.401 ± 0.003	**2.8%**

Table 3: Experimental results on varying observation and forecasting ranges for the medical datasets. Evaluation measure is MSE. Lower is better. Best results are in bold. ME indicates memory error.

CRU, Neural Flow, and GRU-ODE-Bayes. We evaluate them in terms of both execution time (batch size: 64) and MSE. The results, presented in Figure 4, show that for datasets with longer time series like MIMIC-IV and USHCN, GraFITi significantly outpaces ODE and flow-based models. Specifically, GraFITi is over 5 times faster than the fastest ODE model, LinODEnet. Even for shorter time series datasets like Physionet'12 and MIMIC-III, GraFITi remains twice as fast as LinODEnet.

Varying observation and forecast ranges This experiment is conducted with two different observation ranges (24 and 36 hours) and two different prediction ranges for each observation range. For the observation range of 24 hours, the prediction ranges are 12 and 24 hours, and for the observation range of 36 hours, the prediction ranges are 6 and 12 hours. The results are presented in Table 3. Again GraFITi is the top performer, followed by LinODENet. Significant gains in forecasting accuracy are observed in the MIMIC-III and MIMIC-IV datasets. *On average, GraFITi improves the accuracy of LinODEnet, the next best IMTS forecasting model, by* 8.5% *in MIMIC-III,* 15.5% *in MIMIC-IV, and* 2.6% *in the Physionet'12 dataset*. Models avoiding missing value indicators perform best in the related tasks such as IMTS interpolation and classification (Horn et al. 2020; Shukla and Marlin 2021, 2022; Yalavarthi, Burchert, and Schmidt-Thieme 2023, 2022), we see the similar trend for forecasting. Additionally, the Sparsity Structure Graph improves model performance as, by design, it allows inductive bias of the data flows through the model.

6.5 Limitations

The GraFITi model is a potential tool for IMTS forecasting. However, GraFITi faces a challenge when applied to Asynchronous Time Series (AsTS) datasets. In such datasets, channels are observed asynchronously at various time points, resulting in disconnected sparse graphs. This disconnection hinders the flow of information and can be problematic when channels have a strong correlation towards the forecasts as model may not be able to capture these correlations. It can be seen from Table 4 where GraFITi is compared with the next best baseline model LinODENet for varying sparsity levels using MIMIC-III dataset. The performance of GraFITi deteriorates with increase in sparsity levels and gets worst when the series become asynchronous. The breaking point is dataset-specific, as the model breaks when the graph is disconnected.

Model	IMTS	AsTS	AsTS +10%	AsTS +50%	AsTS +90%
GraFITi	**0.396**	0.931	0.845	**0.547**	**0.413**
LinODENet	0.446	**0.894**	**0.815**	0.581	0.452

Table 4: GraFITi with varying sparsity levels on MIMIC-III dataset. 'IMTS' dataset refers to the actual dataset, 'AsTS' is a synthetic asynchronous time series dataset created by observing only 1 channel at each time point. 'AsTS + x%' is created by retrieving x% of the missing observations. Goal is to observe 36 hours and forecast the next 3 time steps.

Moreover, the existing model cannot handle meta data associated with the IMTS. One possible solution to both the challenges is to interconnect all the channel nodes including meta data (assuming it as additional channel) if present, and apply a distinct multi-head attention on them. This will help to share the information among all the nodes even in asynchronous setup. In future, we aim to enhance GraFITi to handle AsTS datasets and meta data.

7 Conclusions

In this paper, we propose a Graph based model called GraFITi for the forecasting of irregularly sampled time series with missing values (IMTS). First, we represent the time series as a Sparsity Structure Graph with channels and observation times as nodes and observation measurements as edges; and re-represent the task of time series forecasting as an edge weight prediction problem in a graph. An attention based architecture is used for learning the interactions between the nodes and edges in the graph. We experimented on 4 datasets including 3 real world and 1 synthetic dataset for various observation and prediction ranges. The extensive experimental evaluation demonstrates that the proposed GraFITi provides superior forecasts compared to the state-of-the-art IMTS forecasting models.

Acknowledgments

This work was supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK), Germany, within the framework of the IIP-Ecosphere project (project number: 01MK20006D); co-funded by the Lower Saxony Ministry of Science and Culture under grant number ZN3492 within the Lower Saxony "Vorab" of the Volkswagen Foundation and supported by the Center for Digital Innovations (ZDIN); and also by the German Federal Ministry of Education and Research (BMBF) through the Program "International Future Labs for Artificial Intelligence" (Grant 1DD20002A – KIWI-BioLab)"

References

Ahmed, N.; and Schmidt-Thieme, L. 2023. Structure-aware decoupled imputation network for multivariate time series. *Data Mining and Knowledge Discovery*, 1–21.

Baytas, I. M.; Xiao, C.; Zhang, X.; Wang, F.; Jain, A. K.; and Zhou, J. 2017. Patient subtyping via time-aware LSTM networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 65–74.

Biloš, M.; Sommer, J.; Rangapuram, S. S.; Januschowski, T.; and Günnemann, S. 2021. Neural Flows: Efficient Alternative to Neural ODEs. *Advances in Neural Information Processing Systems*, 34: 21325–21337.

Cao, D.; Wang, Y.; Duan, J.; Zhang, C.; Zhu, X.; Huang, C.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in Neural Information Processing Systems*, 33: 17766–17778.

Cao, W.; Wang, D.; Li, J.; Zhou, H.; Li, L.; and Li, Y. 2018. Brits: Bidirectional recurrent imputation for time series. *Advances in Neural Information Processing Systems*, 31.

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1): 1–12.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31.

Chen, Y.; Segovia, I.; and Gel, Y. R. 2021. Z-GCNETs: time zigzags at graph convolutional networks for time series forecasting. In *International Conference on Machine Learning*, 1684–1694. PMLR.

Cini, A.; Marisca, I.; and Alippi, C. 2022. Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks. In *International Conference on Learning Representations*.

De Brouwer, E.; Simm, J.; Arany, A.; and Moreau, Y. 2019. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. *Advances in Neural Information Processing Systems*, 32.

De Gooijer, J. G.; and Hyndman, R. J. 2006. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3): 443–473.

De Sá, H. R.; and Prudêncio, R. B. 2011. Supervised link prediction in weighted networks. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, 2281–2288. IEEE.

Fu, C.; Zhao, M.; Fan, L.; Chen, X.; Chen, J.; Wu, Z.; Xia, Y.; and Xuan, Q. 2018. Link weight prediction using supervised learning methods and its application to yelp layered network. *IEEE Transactions on Knowledge and Data Engineering*, 30(8): 1507–1518.

Horn, M.; Moor, M.; Bock, C.; Rieck, B.; and Borgwardt, K. 2020. Set functions for time series. In *International Conference on Machine Learning*, 4353–4363. PMLR.

Hou, Y.; and Holder, L. B. 2017. Deep learning approach to link weight prediction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 1855–1862. IEEE.

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

Johnson, A. E.; Pollard, T. J.; Shen, L.; Lehman, L.-w. H.; Feng, M.; Ghassemi, M.; Moody, B.; Szolovits, P.; Anthony Celi, L.; and Mark, R. G. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1): 1–9.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Krishnan, R.; Shalit, U.; and Sontag, D. 2017. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Krishnan, R. G.; Shalit, U.; and Sontag, D. 2015. Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

Kumar, S.; Spezzano, F.; Subrahmanian, V.; and Faloutsos, C. 2016. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 221–230. IEEE.

Li, S. C.-X.; and Marlin, B. M. 2015. Classification of Sparse and Irregularly Sampled Time Series with Mixtures of Expected Gaussian Kernels and Random Features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, 484–493.

Lim, B.; and Zohren, S. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194): 20200209.

Lipton, Z. C.; Kale, D.; and Wetzel, R. 2016. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Machine Learning for Healthcare Conference*, 253–270. PMLR.

Marisca, I.; Cini, A.; and Alippi, C. 2022. Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations. In *The First Learning on Graphs Conference*.

Menne, M. J.; Williams Jr, C.; and Vose, R. S. 2015. United States historical climatology network daily temperature, precipitation, and snow data. *Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, Oak Ridge, Tennessee*.

Murata, T.; and Moriyasu, S. 2007. Link prediction of social networks based on weighted proximity measures. In

IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), 85–88. IEEE.

Rubanova, Y.; Chen, R. T.; and Duvenaud, D. K. 2019. Latent ordinary differential equations for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32.

Satorras, V. G.; Rangapuram, S. S.; and Januschowski, T. 2022. Multivariate time series forecasting with latent graph inference. *arXiv preprint arXiv:2203.03423*.

Schirmer, M.; Eltayeb, M.; Lessmann, S.; and Rudolph, M. 2022. Modeling Irregular Time Series with Continuous Recurrent Units. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, 19388–19405. PMLR.

Scholz, R.; Born, S.; Duong-Trung, N.; Cruz-Bournazou, M. N.; and Schmidt-Thieme, L. 2023. Latent Linear ODEs with Neural Kalman Filtering for Irregular Time Series Forecasting.

Shukla, S. N.; and Marlin, B. 2021. Multi-Time Attention Networks for Irregularly Sampled Time Series. In *International Conference on Learning Representations*.

Shukla, S. N.; and Marlin, B. 2022. Heteroscedastic Temporal Variational Autoencoder For Irregularly Sampled Time Series. In *International Conference on Learning Representations*.

Silva, I.; Moody, G.; Scott, D. J.; Celi, L. A.; and Mark, R. G. 2012. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, 245–248. IEEE.

Tashiro, Y.; Song, J.; Song, Y.; and Ermon, S. 2021. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34: 24804–24816.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *stat*, 1050: 20.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. 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*, 753–763.

Yalavarthi, V. K.; Burchert, J.; and Schmidt-Thieme, L. 2022. DCSF: Deep Convolutional Set Functions for Classification of Asynchronous Time Series. In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*, 1–10. IEEE.

Yalavarthi, V. K.; Burchert, J.; and Schmidt-Thieme, L. 2023. Tripletformer for Probabilistic Interpolation of Irregularly sampled Time Series. In *2023 IEEE International Conference on Big Data (BigData)*, 986–995. IEEE.

Yang, X.; and Wang, B. 2020. Local ranking and global fusion for personalized recommendation. *Applied Soft Computing*, 96: 106636.

You, J.; Ma, X.; Ding, Y.; Kochenderfer, M. J.; and Leskovec, J. 2020. Handling missing data with graph representation learning. *Advances in Neural Information Processing Systems*, 33: 19075–19087.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2022. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*.

Zhao, J.; Miao, L.; Yang, J.; Fang, H.; Zhang, Q.-M.; Nie, M.; Holme, P.; and Zhou, T. 2015. Prediction of links and weights in networks by reliable routes. *Scientific reports*, 5(1): 12261.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 27268–27286. PMLR.

Zulaika, U.; Sanchez-Corcuera, R.; Almeida, A.; and Lopez-de Ipina, D. 2022. LWP-WL: Link weight prediction based on CNNs and the Weisfeiler–Lehman algorithm. *Applied Soft Computing*, 120: 108657.