# GLIMA: Global and Local Time Series Imputation with Multi-directional Attention Learning

Qiuling Suo*, Weida Zhong*, Guangxu Xun†, Jianhui Sun†, Changyou Chen*, Aidong Zhang†
*Department of Computer Science and Engineering, State University of New York at Buffalo, NY, USA
Email: {qiulings, weidazho, changyou}@buffalo.edu
†Department of Computer Science, University of Virginia, VA, USA
Email: {gx5bt, js9gu, aidong}@virginia.edu

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9378408

***Abstract*—Missing data, which commonly appears in multivariate time series, has been widely recognized as a key challenge in time series analysis. Many commonly used imputation methods either ignore the temporal dependencies of time series data, or do not adequately utilize the relationships among variables. State-of-the-art methods on time series imputation are built on Recurrent Neural Networks (RNNs), which utilize the historical information to estimate current values sequentially. However, RNNs rely heavily on the output of nearby timestamps, which may lead to important information lost for long sequences. Moreover, individual variables typically present different dynamics and missingness patterns, which is neglected by the global RNN hidden states. In this paper, we propose an imputation framework to learn both global and local dependencies of multivariate time series, as well as a multi-dimensional self-attention to learn capture distant correlations across both time and feature. Extensive experiments show that the proposed framework outperforms the state-of-the-art methods in the imputation task, and benefits the downstream task.**

***Index Terms*—Time Series, Missing Data, Recurrent Imputation, Self-Attention.**

## I. Introduction

With the advances in data collection technologies, large amounts of electronic data, especially the multivariate time series, have emerged. Multivariate time series are ubiquitous in many real world applications, including healthcare prediction [1], weather forecasting [2], financial marketing [3], etc. Nevertheless, the data inevitably carries missing values due to various reasons such as sensor damage, data corruption, merging irregular sampled data, and human mistakes in recording. The missingness is a common issue and poses a fundamental challenge to researchers and engineers for analyzing multivariate time series.

Since many machine learning algorithms often require complete datasets, the missingness hinders the advanced analysis of time series. Therefore, a fundamental task in time series data mining is to fill in the missing data with reasonable values. Many methods have been developed on the time series imputation problem. Traditional imputation methods such as K-nearest neighbor (KNN) [4], Matrix Factorization [5] and Multivariate Imputation by Chained Equations (MICE) [6] ignore the temporal relationships among time points. Statistical time series models such as dynamic linear models (DLM) [7] suffer from the incapability of modeling large-scale complex time series. Recently, Recurrent Neural Network (RNN) based methods [1], [8]–[12] have been developed to capture the longitudinal temporal correlations and cross-sectional feature correlations for time series imputation. Meanwhile, RNNs have been incorporated into Generative Adversarial Network (GAN) to generate complete time series [13], [14]. Current RNN-based imputation methods sequentially estimate missing values as the prediction of the recurrent system over time, and have achieved state-of-the-art performances. However, there are still some practical issues of these methods.

Firstly, it is difficult for a global RNN to capture the peculiarity of individual series, when various series are heterogeneous. Typically, values at each timestamp in the multivariate time series are fed into an RNN cell as the input vector, and a hidden state is obtained to represent the mixed multi-variables. However, the per-series information, such as variation in scales, dynamic patterns and the irregularity in missingness patterns, is implicitly neglected by the global model, which may restrict the imputation performance.

Secondly, RNN may not fully capture the correlations of distance variables. The distant correlations can exist in both temporal and cross-feature dimensions, and can provide important information to estimate the current values. For example, temperature is usually higher in summer and lower in winter, and the repetition of this similar patterns can help to predict the temperature values in later temporal cycles; as affected by the temperature, the heating cost is expected to recur every year around the same time. Self-attention provides a flexible way to select and represent information of time series, and is complementary to RNN based models. However, using self-attention alone may not be sufficient, as it is difficult to obtain accurate attention weights on data with heavy missingness.

In this paper, we propose a **G**lobal and **L**ocal **I**mputation with **M**ulti-directional **A**ttention model (**GLIMA**) that focuses on learning both global dependencies and local properties of multivariate time series. In the proposed framework, an efficient way of calculating sequential patterns of individual variables is developed, namely, local imputation. In this way, the short-term correlations between RNN steps can be fully captured. The weighted sum of the short-term states along with current hidden states are concatenated to perform the historical imputation. Meanwhile, we develop a multi-directional self-attention method on top of the RNNs to capture long-term dependencies. The multi-directional self-attention method directly measures the correlations between data entries across both time and feature dimensions. We then concatenate the representations from the two self-attentions, and map it to the original space to obtain the self-attentive imputation. The RNNs and multi-directional self-attention modules are optimized jointly. Our main contributions can be summarized as follows,

- We propose a novel framework that enhances both short-term correlations and long-term correlations for imputing multivariate time series with missing values. The framework captures global and local information from times series, as well as distant dependencies.
- Besides capturing short-term enhanced global information, we explore the structure of RNN to enable local per-series dynamics to be efficiently captured by variable-wise hidden states, which facilitates the imputation task.
- A multi-directional attention mechanism is employed to directly capture the dependencies between data entries across both time and feature dimensions, which enables that the long-term dependencies can be better captured.
- Comparing with various state-of-the-art imputation approaches, our method learns accurate estimated values under different missing scenarios, and benefits the downstream classification task.

## II. Related Work

The past decades have witnessed a large body of imputation approaches to deal with the missing data issue with time series. These methods can be roughly divided into statistical imputation, conventional machine learning based imputation and deep learning based imputation.

The replacement method fills the missing values with some statistical attributes, such as mean, most common value or last observed valid value. Although straightforward to implement, the simple replacement fails to capture the relationship between imputed variables and observed variables. Classical statistical time series models such as autoregressive (AR) models and dynamic linear models (DLM) [7] model the temporal dependencies, but they are essentially linear and may not be suitable for modern complex large-scale data. Machine learning based imputation methods such as K-nearest neighbors (KNN) [4], Matrix Factorization [5] and Multivariate Imputation by Chained Equations (MICE) [6] do not explicitly incorporate temporal dependencies. Recent work extends the conventional imputation methods by modeling temporal dependencies. [15] developed a 3D-MICE model to combine MICE with Gaussian process, and [16] regularized matrix factorization using autoregressive models.

Deep learning models, especially RNNs, which are capable of handing temporal dependencies of sequential observations, have been exploited in time series imputation recently. Based on Gated Recurrent Unit (GRU) [17], [1] developed GRU-D, which represents a missing value as the combination of the last observed value and the global mean. [18] proposed temporal belief memory (TBM), which computes a belief of the last observation over time for each variable and imputes a missing value based on that individual belief in both forward and backward directions. [8] and [9] estimated the missing values in a bidirectional recurrent dynamical system and across variables. Another trend is to combine GAN [19] and RNN to reconstruct realistic time series from a low-dimensional vector. The GAN-based work includes a two-stage model [13] and an end-to-end one [14].

Self-attention [20] has attracted enormous interest and achieved great success in modeling texts, due to its highly parallelizable computation and flexibility in modeling dependencies. In very recent work, self-attention mechanism is employed in modeling time series [21]–[23]. However, relying purely on self-attention may not be sufficient on data with missing values, as the attention weights may not be learned accurately, especially under high missing rate. Therefore, we build the self-attention layer on top of the recurrent imputation process, in order to obtain the complement inputs for attention calculation.

Considering capturing temporal dependencies in time series, our method is also related to the forecasting approaches [24]–[28]. In particular, there are other approaches that augment a global time series model with local parameters [29]–[31] and local models for multi-variables [32]. However, these models focus on predicting future values of a target series using exogenous time series, and cannot be directly used in our problem that the input time series has numbers of missing values. The densely connected network for short-term enhancement is inspired by DenseNet [33], but there are key differences. [33] is designed to alleviate the vanishing-gradient problem for deep architectures, but our method is to alleviate the inaccuracy of imputed values and better learn relationships between sequential time points.

## III. Methodology

In this section, we introduce the details of our proposed method. We first formulate the problem and introduce the basic notations. Then, we introduce the two main parts of proposed method: global and local RNNs for recurrent imputation, and hybrid self-attention.

### *A. Overall Architecture*

A multivariate time series $\mathbf{X}$ can be viewed as a matrix, with $T$ observations[1] and $D$ variables, and denoted as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]^\top$, where $\mathbf{x}_t = [x_t^1, x_t^2, ..., x_t^D]^\top \in \mathbb{R}^D$ represents the vector of variables at time $t$. The $d$-th series $\mathbf{x}^d = [x_1^d, x_2^d, ..., x_T^d]^\top \in \mathbb{R}^T$ stands for the $d$-th variable over time. Since $\mathbf{X}$ carries missing values, we introduce a masking matrix $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_T]^\top \in \mathbb{R}^{T\times D}$, where $\mathbf{m}_t \in \{0,1\}^D$ denotes which variables are missing at time stamp $t$: if $x_t^d$ is missing, $m_t^d = 0$; otherwise $m_t^d = 1$. The purpose of multivariate time series imputation is to impute the missing values in $\mathbf{X}$ as accurately as possible. The overall framework is shown in Figure 1. We first use a global RNN and a set of local RNNs to learn the short-term temporal

[1]The length of each series is not necessarily the same. For notational convenience, we assume the time dimension is $T$ for all series.

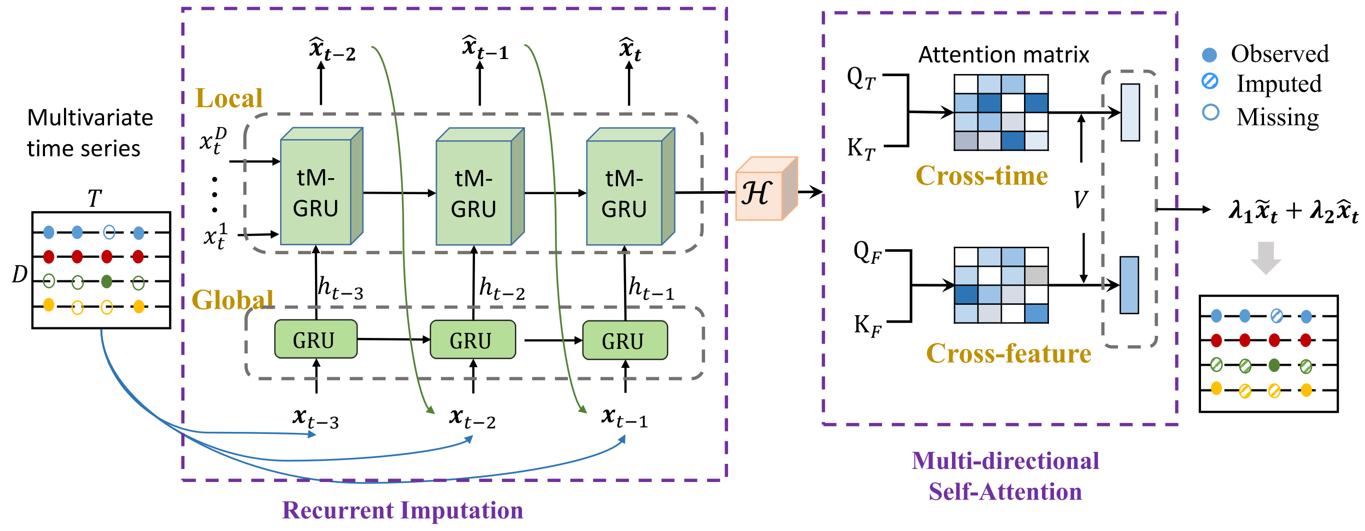


Fig. 1: Overall framework. The input data with missing values is first sequentially imputed through global and local RNNs in the recurrent imputation process, and then fed into a multi-directional self-attention layer. The final imputed value is the combination of estimations obtained by recurrent layers and attention layer.

relationships sequentially, and then use a hybrid multi-head self-attention layer to learn long-term correlations across time and features.

## *B. Recurrent Imputation*

We use a global GRU to learn the mixed information of all variables, and a set of local GRUs to capture different dynamics of individual variables.

*1) Global Model:* We feed the input vector $\mathbf{x}_t$ at timestamp $t$ into an RNN to obtain a hidden representation which contains the mixed information of $D$ variables. A time-decayed GRU variant in [13] is adopted to sequentially process the incomplete time series. Supposing $\delta_t \in \mathbb{R}^D$ is the time interval between two adjacent existent values, a temporal decay factor $\gamma_t \in \mathbb{R}^p$ is introduced as,

$$\gamma_t = \exp\{-\max(0, \mathbf{W}_\gamma^\top \delta_t + \mathbf{b}_\gamma)\}, \tag{1}$$

where $\mathbf{W}_\gamma \in \mathbb{R}^{D\times p}$ and $\mathbf{b}_\gamma \in \mathbb{R}^p$ are learnable parameters, and $p$ is the predefined size of hidden vectors. Then the time decayed hidden state $\mathbf{h}'_{t-1} \in \mathbb{R}^p$ can be obtained by $\mathbf{h}'_{t-1} = \gamma_t \circ \mathbf{h}_{t-1}$, where $\circ$ denotes the element-wise multiplication. Based on the hidden state $\mathbf{h}'_{t-1}$, we can obtain an estimated vector $\hat{\mathbf{x}}_{gt}$ for the next timestamp $t$ by linear regression as,

$$\hat{\mathbf{x}}_{gt} = \mathbf{W}_g \mathbf{h}'_{t-1} + \mathbf{b}_g, \tag{2}$$

where $\mathbf{W}_g \in \mathbb{R}^{D\times p}$ and $\mathbf{b}_g \in \mathbb{R}^D$ are model parameters to be learned. The next hidden state $\mathbf{h}_t$ can be obtained by,

$$\mathbf{x}_{gt}^c = \mathbf{m}_t \circ \mathbf{x}_t + (1-\mathbf{m}_t) \circ \hat{\mathbf{x}}_{gt}, \tag{3}$$
$$\mathbf{h}_t = \mathrm{GRU}(\mathbf{h}'_{t-1}, \mathbf{x}_{gt}^c), \tag{4}$$

where $\mathbf{x}_{gt}^c$ is a complement vector with which the missing values in $\mathbf{x}_t$ are replaced by the corresponding estimated values in $\hat{\mathbf{x}}_{gt}$. The above process is performed recurrently, and we can obtain a set of hidden states that contain historical information of multiple variables.

*2) Short-term Enhancement:* Instead of directly using the above global RNN to perform imputation sequentially, we use a densely-connected RNN, namely denseRNN, to enhance short-path connections between current timestamp and the previous ones. Inspired by [33] which connects each layer to every other layer in a deep neural network, we connect the hidden states of timestamps to allow useful information flow through shortcuts. To enhance the short-term correlations, we first obtain a vector $\mathbf{u}_t$ containing a collection of historical hidden states within a window,

$$\mathbf{u}_t = f([\mathbf{h}_{t-k}; \mathbf{h}_{t-k+1}; ...; \mathbf{h}_{t-1}]), \tag{5}$$

where $f(\mathbf{x}) = \mathbf{W}_u^\top \mathbf{x} + b_u$ is a transformation function, $\mathbf{u}_t \in \mathbb{R}^q$, with the learnable parameters $\mathbf{W}_u \in \mathbb{R}^{kp\times q}$ and $b_u \in \mathbb{R}^{kp}$. Then the combined information can be obtained as,

$$\widehat{\mathbf{h}}_t = [\mathbf{h}_t; \mathbf{u}_t], \tag{6}$$

where $\widehat{\mathbf{h}}_t$ contains information from the current state and the weighted sum of previous $k$ states. After obtaining the historical latent representation, we use a fully connected layer to predict variable values,

$$\widetilde{\mathbf{x}}_{gt} = \mathbf{W}_h^\top \widehat{\mathbf{h}}_{t-1} + \mathbf{b}_h, \tag{7}$$

where $\mathbf{W}_h \in \mathbb{R}^{(p+q)\times D}$ and $\mathbf{b}_h \in \mathbb{R}^D$ are model parameters to be learned. In this way, we can obtain the estimation $\widetilde{\mathbf{x}}_{gt}$ using historical observations. In practice, $k$ is chosen to be small ($<= 3$), as the information from a relatively far timestamp in the short-term range tend to be less important. In our method, the short-term enhanced estimation $\widetilde{\mathbf{x}}_{gt}$ is used as the global imputation value, in substitute of $\hat{x}_{gt}$. In the following sections, $\widetilde{\mathbf{x}}_{gt}$ and $\hat{x}_{gt}$ are used interchangeably.

The proposed denseRNN should not be confused with performing attention [34] on hidden states. DenseRNN enhances the dependencies among states within a short-term range. In contrast, attention mechanism is performed on all the historical states, which may import noisy information if some long-term states are not that important. Moreover, instead of using conventional attention mechanism on hidden states, in subsection III-C, we will introduce a hybrid self-attention mechanism learns the long-term dependencies directly from the observed space.

*3) Local Models:* In multivariate time series, different time series may exhibit different patterns, e.g., in the lab measurements of human health, some variables remain stable during a long time interval on account of the homeostatic properties of human body; other variables however may change dramatically due to the status of diseases. Using an RNN with mixed variables as inputs may not sufficiently capture the dynamics of individual variables. Therefore, in addition to the global imputation, we use a set of $\{\mathrm{RNN}_d\}_{d=1}^{D}$ to capture the temporal information for each series separately. However, calculating the set of local RNNs can be time consuming, especially when there are a large number of variables. To overcome this issue, inspired by the forecasting model in [32], we propose a time-decayed multivariate GRU cell, named as tMGRU, to learn variable-wise hidden states for time series with missing values.

Given a hidden state matrix $\mathcal{H}_t$ which represents individual variables at timestamp $t$, we denote it as $\mathcal{H}_t = [\boldsymbol{h}_t^1, ..., \boldsymbol{h}_t^D] \in \mathbb{R}^{D\times p_s}$ with $\boldsymbol{h}_t^d \in \mathbb{R}^{p_s}$ representing the hidden state of $d$-th variable, and $p_s$ the dimension of the hidden vector. Similar to the global imputation, we first obtain a temporal decay factor $\gamma_{st} \in \mathbb{R}^{D\times p_s}$ indicating the patterns of consecutive time intervals for individual variables, and then calculate the time decayed hidden state matrix $\mathcal{H}'_{t-1} = \gamma_{st} \circ \mathcal{H}_{t-1}$. Since the input vector $\mathbf{x}_t$ contains missing values, which is harmful to the sequential learning process, we estimate the missing values for each variable using its hidden representation as,

$$\hat{\mathbf{x}}_{st} = \boldsymbol{\mathcal{W}}_s \circledast \mathcal{H}_t + \mathbf{b}_s, \tag{8}$$

where $\boldsymbol{\mathcal{W}}_s \in \mathbb{R}^{D\times p_s}$ and $\mathbf{b}_s \in \mathbb{R}^{D}$ are the learnable parameters, $\circledast$ is the tensor dot operation and $\hat{\mathbf{x}}_{st} \in \mathbb{R}^{D}$ is the estimated vector at time $t$. Similarly to Eq. (3), we can obtain a complement vector $\mathbf{x}_{st}^c$ by replacing the missing values in $\mathbf{x}_t$ using the estimated values in $\hat{\mathbf{x}}_{st}$. Utilizing the global information obtained in Eq. (4), we concatenate $\mathbf{h}_t$ with each element of $\mathbf{x}_{st}^c$, and use the obtained vector as the input of tMGRU. With the hidden state matrix $\mathcal{H}'_{t-1}$ and complement input $\mathbf{x}_{st}^c$, we can obtain the next hidden state matrix $\mathcal{H}_t$ using,

$$\begin{aligned}
\gamma_t =& \exp\{-\max(0, \mathbf{W}_{d\gamma}\delta_t + \mathbf{b}_\gamma)\}, \quad \mathcal{H}'_{t-1} = \gamma_t \circ \mathcal{H}_{t-1},\\
&\mathbf{z}_t = \sigma(\boldsymbol{\mathcal{W}}_z \circledast \mathcal{H}'_{t-1} + \boldsymbol{\mathcal{U}}_z \circledast [\mathbf{x}_{st}^c; \mathbf{h}_{t-1}] + \mathbf{b}_z),\\
&\mathbf{r}_t = \sigma(\boldsymbol{\mathcal{W}}_r \circledast \mathcal{H}'_{t-1} + \boldsymbol{\mathcal{U}}_r \circledast [\mathbf{x}_{st}^c; \mathbf{h}_{t-1}] + \mathbf{b}_r),\\
&\tilde{\mathcal{H}}_t = \tanh(\mathbf{r}_t \circ \boldsymbol{\mathcal{W}}_h \circledast \mathcal{H}'_{t-1} + \boldsymbol{\mathcal{U}}_h \circledast [\mathbf{x}_{st}^c; \mathbf{h}_{t-1}] + \mathbf{b}_h),\\
&\mathcal{H}_t = \mathbf{z}_t \circ \mathcal{H}'_{t-1} + (1 - \mathbf{z}_t) \circ \tilde{\mathcal{H}}_t,
\end{aligned} \tag{9}$$

where $\sigma(\cdot)$ is the activation function, $\mathbf{h}_t$ is the hidden state obtained from global RNN in Eq. (4), $\mathbf{W}_{d\gamma} \in \mathbb{R}^{D_s}$, $\boldsymbol{\mathcal{W}}_z$,$\boldsymbol{\mathcal{W}}_r$, $\boldsymbol{\mathcal{W}}_h \in \mathbb{R}^{D\times p_s\times p_s}$ and $\boldsymbol{\mathcal{U}}_z$,$\boldsymbol{\mathcal{U}}_r$,$\boldsymbol{\mathcal{U}}_h \in \mathbb{R}^{D\times p_s\times 1}$ are learnable parameters. The terms $\boldsymbol{\mathcal{W}}_{z,r,h} \circledast \mathcal{H}_{t-1}$ and $\boldsymbol{\mathcal{U}}_{z,r,h} \circledast \mathbf{x}_t$ capture the update from last hidden state and the current input respectively.

The tensor dot operation $\circledast$ is the product of two tensors along the $D$ axis, e.g., $\boldsymbol{\mathcal{W}} \circledast \mathcal{H}_{t-1} = [\mathbf{W}^1\boldsymbol{h}_t^1, ..., \mathbf{W}^D\boldsymbol{h}_t^D]^\top$ where $\mathbf{W}^d\boldsymbol{h}_t^d \in \mathbb{R}^{p_s}$. The tensor product makes variable-wise GRUs to be learned in a parallel manner, which improves the training efficiency. The tMGRU calculates a set of independent GRUs, each of which processes the parameters for one series. With this process, we can obtain a group of representations $\{\boldsymbol{h}_t^d\}_{d=1}^{D}$ for different variables.

Given the global representation $\mathbf{h}_t$ and variable-specific representations $\mathcal{H}_t$, we integrate them to obtain a weighted estimation by,

$$\hat{\mathbf{x}}_t = \alpha_1\hat{\mathbf{x}}_{gt} + \alpha_2\hat{\mathbf{x}}_{st}, \tag{10}$$

where $\alpha_1 = softmax(\mathbf{W}_{rg}^\top\mathbf{h}_t/(\mathbf{W}_{rg}^\top\mathbf{h}_t + \mathbf{W}_{rs}^\top\boldsymbol{h}_t^d))$ with $\mathbf{W}$ to be learned. $\alpha_2$ is defined in a similar way. In this way, we can obtain the historical imputation $\hat{x}$ that can fully utilize the multivariate information and variable-specific characteristics. In practice, we use bi-directional global and local RNNs to capture the dependencies from past and future timestamps. In each timestamp, we use $\hat{\mathbf{x}}_t$ obtained in Eq. (10) in substitute of $\hat{\mathbf{x}}_{gt}$ in Eq. (3) and $\hat{\mathbf{x}}_{st}$ in Eq. (9) to obtain a more accurate complement vector as input of the next timestamp.

## C. Multi-directional Self-Attention

Although the aforementioned global and local recurrent models for imputation are capable of utilizing short-term memories, they cannot effectively capture the long-term correlations. Therefore, we further propose to conduct direct connections between two arbitrary variables on top of recurrent imputation. Previously, self-attention mechanism [20] has been employed to capture long-term correlations between feature vectors across time [23]. However, this time-level attention mixes the information from multivariate series and ignores the correlations between individual series, which may not fully reveal the dynamic patterns of individual variables. Therefore, we propose a multi-directional self-attention mechanism, so that distant correlations of individual values can be captured.

The proposed self-attention layer contains two blocks, i.e., time-level attention and variable-level attention. With the hidden state tensor $\mathcal{H} = \{\mathcal{H}_t\}_{t=1}^{T} \in \mathbb{R}^{T\times D\times p_s}$ obtained in Section III-B, we map the it into queries, keys and values through linear embedding operations. For time-level attention, we obtain $\mathbf{Q}_T \in \mathbb{R}^{T \times d_t}$ and $\mathbf{K}_T \in \mathbb{R}^{T \times d_t}$, and for variable-level attention, we obtain $\mathbf{Q}_F \in \mathbb{R}^{D \times d_f}$ and $\mathbf{K}_F \in \mathbb{R}^{D \times d_f}$, where $d_t$ and $d_f$ are the embedding dimensions. The value $\mathbf{V} \in \mathbb{R}^{T \times D \times d_v}$ is shared between two attention blocks. Thus, the attention map $\mathbf{A}_T$ can be obtained by,

$$\mathbf{A}_T(\mathcal{H}) = Softmax(\frac{\mathbf{Q}_T \mathbf{K}_T^\top}{\sqrt{d_t}}), \tag{11}$$

where $\mathbf{A}_T$ is a $T \times T$ matrix whose entry $(i, j)$ represents the strength of connection between vectors $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ in $\mathcal{H}$. The output of self-attention is the weighted sum of values,

$$\mathbf{O}_T(\mathbf{Q}_T, \mathbf{K}_T, \mathbf{V}) = \mathbf{A}_T \mathbf{V}, \tag{12}$$

where $\mathbf{O}_T \in \mathbb{R}^{T \times D \times d_v}$. Following a similar procedure, we can obtain the feature-level attention map $\mathbf{A}_F$ by calculating the dot product of $\mathbf{Q}_F$ and $\mathbf{K}_F$ followed by the rescaling and softmax operations. After that, we can obtain the variable-level attentional output $\mathbf{O}_F = \mathbf{A}_F \mathbf{V} \in \mathbb{R}^{T \times D \times d_v}$.

In practice, we employ the multi-head attention $(MH)$ [20], which maps the query, key and value vectors into different subspaces and is expected to help the model capture different semantic correlations. Assuming we have $h$ subspaces, we will obtain $h$ output vectors. We can concatenate those output vectors and map it into the final output vector via a linear projection,

$$\begin{gathered} MH(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concate(head_1, ..., head_h)\mathbf{W}^O, \\ \text{where } head_i = \mathbf{A}ttn(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \end{gathered} \tag{13}$$

where $\mathbf{W}^O$, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ are the parameters of linear projections. We apply the multi-head self-attention on time and feature levels simultaneously, and obtain both time-time and feature-feature relationships. The two attentional outputs $\mathbf{O}_T$ and $\mathbf{O}_F$ are then combined via a linear mapping layer to obtain a hybrid representation,

$$\mathbf{O}_{TF} = \mathbf{W}_{TF}^\top [\mathbf{O}_T; \mathbf{O}_F] + \mathbf{b}_{TF}, \tag{14}$$

where $\mathbf{W}_{TF} \in \mathbb{R}^{2d_v \times d_v}$ and $\mathbf{b}_{TF} \in \mathbb{R}^{d_v}$ are two learnable parameters. The output tensor $\mathbf{O}_{TF} \in \mathbb{R}^{T \times D \times d_v}$ is then mapped to the input space of $\mathbf{X}$ through a multi-layer perceptron,

$$\tilde{\mathbf{X}} = FC(\mathbf{O}_{TF}),$$

where $\tilde{\mathbf{X}} \in \mathbb{R}^{T \times D}$ is the output of the framework, i.e., the attentional estimation of the multivariate time series $\mathbf{x}_t$. The final estimated value $\mathbf{x}_t^e$ is a weighted average of $\hat{\mathbf{x}}_t$ and $\tilde{\mathbf{x}}_t$ whose weights $\lambda_1$ and $\lambda_2$ are calculated using a similar strategy as in Eq. (10). The imputation loss is the mean absolute error (MAE) between the estimated values and observed ones. To accelerate the convergence speed, we accumulate all the estimation errors,

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \mathbf{m}_t \circ (|\mathbf{x}_t - \tilde{\mathbf{x}}_t| + |\mathbf{x}_t - \hat{\mathbf{x}}_t| + |\mathbf{x}_t - \mathbf{x}_t^e|), \tag{15}$$

where $N$ is the number of training data examples, i.e., the number of multivariate time series in the dataset.

## IV. Experiments

In this section, we compare the proposed method with state-of-the-art approaches on a synthetic dataset and three real-world datasets. We investigate three different missing scenarios. The proposed imputation method is evaluated on two tasks: imputation and downstream classification.

### A. Datasets

*1) Synthetic Data:* Based on a similar constructing process introduced in [3], we generate a set of multivariate time series using the seasonal vector autoregression (SVAR(K, k)) equation,

$$\mathbf{y}_t = \left(1 - \sum_{i=1}^{K} \phi_{s,i} B^{si}\right) \sum_{i=1}^{k} \phi_i \mathbf{y}_{t-i} + \sum_{i=1}^{K} \phi_{s,i} \mathbf{y}_{t-si} + \epsilon_t, \tag{16}$$

where $\mathbf{y}_t \in \mathbb{R}^d$ is the vector at time $t$, $\epsilon_t$ is the noise term sampled from a multivariate Gaussian distribution, $\{\phi_i\}_{i=0}^k$ and $\{\phi_{s,i}\}_{i=0}^K \in \mathbb{R}^{d \times d}$ are the coefficients, $s$ denotes the period, $B$ is the backshift operator, and the lags $k$ and $K$ are set to 4 and 2 respectively. We first generate 50 time series as seeds using Eq. (16) with randomly sampled $\{\phi_i\}_{i=0}^k$, $\{\phi_{s,i}\}_{i=0}^K$ and $\epsilon_t$. The length of each time series is 40. We then synthesize 15 time series from each seed by adding different kinds of noise. In this way, each seed generates one cluster/class, and its noisy copies belong to the same cluster/class.

*2) Real-world Data:* We perform experiments on three real-world datasets, including a medical dataset which contains physiological variables, and two urban sensory datasets about air and meteorology, as follows:

- Air Quality Data (Air). This dataset recorded air quality data such as PM2.5 and PM10 hourly. Following [9], we use the PM2.5 measurements from 36 stations in Beijing, and use the PM2.5 values at the 3rd, 6th, 9th, and 12th months as the test data and the other months as the training data. We randomly select 36 consecutive time steps as one time series. The missing pattern is not distributed uniformly.
- KDD CUP 2018 Dataset (KDD18). KDD18 is a public air quality and meteorology dataset that comes from *KDD CUP Challenge 2018*[2]. It contains historical observations from 11 stations in Beijing recorded hourly. We use 11 variables including PM2.5, PM10, temperature, pressure and so on. We split the data for every 48 hours, and obtain 167 data samples.
- PhysioNet Challenge 2012 Dataset (PhysioNet). This is a medical dataset from *PhysioNet Challenge 2012* [35]. It consists of 4,000 multivariate clinical time series from intensive care unit (ICU). Each time series contains 35 measurements such as Albumin, heart-rate, etc. Mortality prediction is performed as a downstream task on this dataset.

[2]http://www.kdd.org/kdd2018

## B. Experimental Setup

*1) Baseline Methods:* We compare our method with various widely-used imputation methods and state-of-the-art methods, including replacement methods, conventional machine learning based and RNN-based methods.

• Replacement Method. We simply replace the missing values using **last** observation before current time, or the global **mean** of corresponding variables.

• **KNN** [4]: For each data sample, we use $k$-nearest neighbor with Euclidean distance to find the similar vectors across different time stamps, and impute the missing values with weighted average of its neighbors.

• **MICE** [6]: Multivariate Imputation by Chained Equations (MICE) fills the missing values by updating regression models iteratively.

• **GRU-D** [1]: It is designed for health-care prediction with missing values. It imputes the missing value with the weighted combination of last observation, global mean, and a recurrent component.

• **TBM** [18]: It bidirectionally updates the imputed values within a reliable time window.

• **M-RNN** [8]: It uses a multi-directional recurrent neural network that interpolates missing values within data streams and imputes across data streams.

• **GAN-I** [13]: It is a two stage model. It uses GAN to learn the distributions of time series, and optimizes the reconstruction loss and adversarial loss. The "noise" input vector is optimized to find the best matched input of the generator.

• **E2GAN** [14]: It is an end-to-end generative model, which impute the incomplete time series by the nearest generated complete time series at one stage. The generator takes a random vector as the input and tries to compress and reconstruct the time series, as well as fooling the discriminator.

• **BRITS** [9]: This method uses bi-directional RNN combined with cross-sectional feature regression to estimate the missing values.

To evaluate the contribution of different components in the proposed framework, we conduct the following ablation study. **GLIMA-g** is the proposed model without global imputation. **GLIMA-l** is without local imputation. Since the hidden states of different series in the global model is mixed, multi-directional self-attention cannot be applied. Therefore, we add only the time-dimensional self-attention on top of the global model to form GLIMA-l. **GLIMA-a** is the model without attention mechanism. Besides, we compare with utilizing self-attention (**Attn**) alone and the missing values are initialized with global mean.

*2) Evaluation:* We evaluate the proposed method on two tasks: imputation and downstream tasks. For the imputation performance, we randomly drop some values from the observed data and use them as the groudtruth values. We measure error between imputed values and the groundtruth in terms of mean absolute error (MAE) and mean relative error (MRE) as defined in [9]. For downstream tasks evaluation, we follow a two-step procedure: first complete the missing values using the imputation methods, and then train a classification model and report the accuracy values. Note that for downstream tasks, we have two types of missing values: intrinsically missing and randomly dropped values.

*3) Implementation Details:* We implement all the RNN-based baselines and the proposed method with PyTorch and keep the same number of hidden units. Adam [36] optimizer is used with the learning rate 0.001. Early stopping strategy is adopted when the validation error does not improve for 10 epochs. For GAN-I and E2GAN, we use the provided Tensorflow source code. The number of parameters in different models are kept roughly the same for fair comparison. The batch size is set to 64. The input datasets are normalized with zero mean and unit variance. We use one attention layer with head number 8, since we empirically find that using more attention layers does not help much in imputation. The non-RNN baselines are implemented using FancyImpute[3] package. For downstream tasks, we use five-fold cross validation to evaluate the classification performance.

## C. Experimental Results

*1) Evaluation on Synthetic Dataset.:* The comparison of imputation and downstream performances on synthetic dataset are shown in Table I and Table II respectively. We compare the results with different missing rates, i.e. 10%, 30% and 50%, which indicate the percentage of values that have been randomly dropped. MAE and MRE are used to evaluate the deviance between imputed values and the ground truths, and smaller MAE/MRE values indicate better performance. With the completed dataset, subsequent downstream analysis can be performed based on the application scenarios. For the synthetic dataset, we evaluate two downstream tasks: classification and clustering. Random forest classifier and K-means clustering algorithm is trained on datasets imputed via different methods. Accuracy and Normalized Mutual Information (NMI) measurements are reported for classification and clustering respectively. Detailed discussions about model comparisons and analysis are provided in the following subsections on real-word datasets.

TABLE I: Performance evaluation for imputation task on synthetic dataset with different missing rates.

| Methods | 10% | | 30% | | 50% | |
|---|---|---|---|---|---|---|
| | MAE | MRE | MAE | MRE | MAE | MRE |
| Last | .8308 | 1.271 | .8149 | 1.259 | .8213 | 1.282 |
| Mean | .6435 | 0.984 | .6399 | 0.989 | .6398 | 0.991 |
| KNN | .6840 | 1.047 | .6697 | 1.035 | .6685 | 1.043 |
| MICE | .6481 | 0.992 | .6483 | 1.002 | .6477 | 1.011 |
| GRU-D | .8032 | 1.239 | .7803 | 1.205 | .7740 | 1.205 |
| TBM | .7464 | 1.151 | .7663 | 1.184 | .7943 | 1.237 |
| M-RNN | .4978 | 0.768 | .5878 | 0.908 | .6110 | 0.951 |
| BRITS | .3265 | 0.503 | .4870 | 0.752 | .5865 | 0.913 |
| Attn | .6312 | 0.973 | .6334 | 0.977 | .6351 | 0.989 |
| GLIMA | **.2751** | **0.421** | **.3636** | **0.562** | **.5609** | **0.876** |

[3]https://github.com/iskandr/fancyimpute

TABLE II: Performance evaluation for classification and clustering tasks on synthetic dataset with different missing rates.

| Methods | 10% | | 30% | | 50% | |
|---|---|---|---|---|---|---|
| | Accu. | NMI | Accu. | NMI | Accu. | NMI |
| Last | .9515 | .9447 | .7747 | .6909 | .5899 | .5325 |
| Mean | .9624 | .9685 | .8573 | .8147 | .6800 | .6433 |
| KNN | .9589 | .9749 | .8104 | .8073 | .6187 | .6527 |
| MICE | .9581 | .9573 | .8277 | .8105 | .6691 | .6468 |
| GRU-D | .9592 | .9541 | .8043 | .7174 | .6264 | .5441 |
| TBM | .9555 | .9645 | .7829 | .7641 | .6184 | .5825 |
| M-RNN | .8152 | .8175 | .5171 | .5967 | .3163 | .4747 |
| BRITS | **.9880** | .9822 | .9216 | .9302 | .6896 | .7042 |
| Attn | .9642 | .9648 | .8376 | .8548 | .6679 | .6954 |
| GLIMA | .9843 | **.9947** | **.9341** | **.9502** | **.7547** | **.7775** |

*2) Imputation Performance Comparison:* We investigate three data missing scenarios, i.e., element-wise missing, block-wise missing and fixed missing values on PhysioNet, KDD18 and Air datasets respectively. For element-wise missing, we randomly remove 10% values of the dataset as the testing data. For block-wise missing, the records from certain sensors are continuously eliminated for a certain period (time length is set to 5). For fixed pattern missing, we follow the strategy of missing value selection used in [37]: finding the missing entries in each month's data, and if entries at the same timestamp are not absent in the next month, their values are used as the ground truth. The comparison of imputation performance on three datasets is shown in Table III. MAE is used to evaluate the deviance between imputed values and the ground truths, and smaller MAE values indicate better performance. We evaluate the performance on normalized space for PhysioNet and KDD18 datasets, and map the data to original space for evaluation on Air dataset.

TABLE III: Performance evaluation for imputation tasks (in MAE).

| | PhysioNet (element-wise) | KDD18 (block-wise) | Air (fixed missing) |
|---|---|---|---|
| Last | 0.4344 | 0.4345 | 45.30 |
| Mean | 0.6910 | 0.5378 | 55.51 |
| KNN | 0.3671 | 0.3290 | 29.79 |
| MICE | 0.5688 | 0.2424 | 27.42 |
| GRU-D | 0.4366 | 0.6467 | 50.80 |
| TBM | 0.4327 | 0.5146 | 45.92 |
| M-RNN | 0.2924 | 0.2669 | 14.13 |
| BRITS | 0.2750 | 0.2178 | 11.05 |
| GAN-I | 0.6470 | 0.9170 | 47.38 |
| E2GAN | 0.7010 | 0.8911 | 45.46 |
| Attn | 0.4481 | 0.4174 | 18.62 |
| GLIMA-a | 0.2705 | 0.2101 | 10.71 |
| GLIMA-g | 0.2893 | 0.2242 | 11.19 |
| GLIMA-l | 0.2830 | 0.2186 | 10.56 |
| GLIMA | **0.2647** | **0.2087** | **10.54** |

As we can see from Table III, the simple replacement methods are inaccurate, as they ignore the statistical irregularities of the data. KNN and MICE take into consideration the relationships among features, but do not model the temporal dependencies explicitly. GRU-D and TBM assume that the missingness is correlated with labels, and impute the missing values implicitly in the process of optimizing classification errors. Since they do not explicitly optimize the imputation error, their performance on the imputation task is not well, but they actually perform very well on classification tasks. M-RNN and BRITS both utilize RNN to capture the temporal dependencies and consider feature correlations at each timestamp, and they can achieve rather good performance. Specifically, since BRITS also learns the relationship among missing values by feeding the complement vector as the input of the RNN cell, the imputed values can be better updated.

Compared to the baselines, the proposed method obtains more accurate imputation results. In our method, global temporal dependencies and local properties of individual series can be fully captured during the recurrent imputation process. By comparison with Attn, we find that utilizing self-attention alone does not perform well, since it needs a good initialization of the missing values to obtain the relationship between query and keys. In the proposed method, self-attention is well initialized through recurrent imputation, so that the time-level and feature-level dependencies can be better learned.

*3) Downstream Task Evaluation:* Completing the missing values enables subsequent analysis of time series. In this subsection, we exploit how imputation accuracy can impact downstream tasks. We follow a two-step procedure: first impute the datasets with different imputation methods, and then train the same classification model with different completed datasets after imputation to evaluate the downstream task performances. We use logistic regression (LR) [38], support vector machine (SVM) [39] with radial basis function (RBF) kernel, random forest (RF) [40] and RNN as the classifier. We compare the classification accuracy on PhysioNet as shown in Table IV. The area under curve (AUC) scores are reported. We can see that RNN-based methods generally outperform conventional imputation methods. The performance of downstream tasks following different imputation methods is generally in accordance with their imputation accuracy, which indicates that the high quality of imputed data benefits the downstream tasks. We also observe that relative performances of some methods between imputation and downstream tasks are not always consistent. In fact, many factors could affect the downstream evaluation. For GAN-I and E2GAN, AUC values in Table IV are reported from the original papers, and the classification methods may have different hyperparameter settings as ours, e.g., number of trees in RF, structure of RNN cell (they use a specifically designed GRUI cell). Moreover, Mean performs worse in imputation, but it can be competitive with other methods in mortality classification on PhysioNet. This may be due to the fact that in healthcare area, missing values tend to be close to some default values such as last observed ones or variable means, and these default values can provide some indication of the mortality probabilities. Note that the purpose of evaluating downstream task is not to find out the best imputation method, but to provide some insight about how an accurate imputation algorithm can be beneficial to downstream tasks.

TABLE IV: Performance evaluation for downstream classification on PhysioNet (in AUC).

| Method | LR | SVM | RF | RNN |
|---|---|---|---|---|
| Last | 0.6935 | 0.8030 | 0.8037 | 0.8177 |
| Mean | 0.7032 | 0.7931 | 0.8137 | 0.8163 |
| KNN | 0.6926 | 0.7961 | 0.8052 | 0.8146 |
| MICE | 0.6438 | 0.7413 | 0.7481 | 0.7790 |
| GRU-D | 0.6796 | 0.8092 | 0.8124 | 0.8192 |
| TBM | 0.6892 | 0.7871 | 0.8026 | 0.8364 |
| M-RNN | 0.7240 | 0.7993 | 0.8052 | 0.8306 |
| BRITS | 0.7262 | 0.8103 | 0.8006 | 0.8551 |
| GAN-I [13] | 0.701 | 0.816 | 0.755 | 0.8603 |
| E2GAN [14] | 0.7955 | 0.820 | 0.7998 | **0.8724** |
| Attn | 0.6815 | 0.7882 | 0.7943 | 0.8045 |
| GLIMA-a | 0.7357 | 0.8224 | 0.8209 | 0.8627 |
| GLIMA-g | 0.6907 | **0.8429** | 0.7969 | 0.8713 |
| GLIMA-l | 0.7194 | 0.8020 | 0.8216 | 0.8489 |
| GLIMA | 0.7684 | 0.8237 | **0.8347** | 0.8692 |

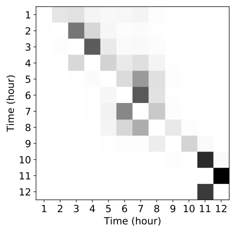
*4) Change of Missing Rates:* To investigate the performance change under different ratios of training and testing data, we change the missing rate from 20% to 70% on KDD18 dataset. The results are shown in Table V. It can be seen that the performance of all methods decreases under high missing rates. This is due to the fact that more data helps to better capture the complex temporal and feature correlations. We evaluate the performance under random element-wise missing in Table V. Compared to block-wise missing in Table III, all methods generally obtain better MAEs, which indicates that block-wise missing is a more difficult problem. Still, the proposed method obtains the best performance.

TABLE V: Imputation evaluation on KDD18 dataset with different element-wise missing rates (in MAE).

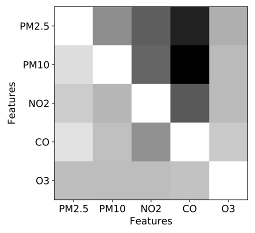
| | 20% | 30% | 40% | 50% | 60% | 70% |
|---|---|---|---|---|---|---|
| Last | .266 | .268 | .302 | .328 | .360 | .408 |
| Mean | .507 | .507 | .508 | .509 | .511 | .515 |
| KNN | .255 | .368 | .282 | .299 | .321 | .355 |
| MICE | .244 | .252 | .261 | .274 | .288 | .310 |
| TBM | .432 | .459 | .447 | .473 | .484 | .509 |
| M-RNN | .290 | .314 | .327 | .338 | .361 | .381 |
| BRITS | .214 | .221 | .230 | .240 | .253 | .270 |
| GAN-I | .736 | .859 | .793 | .847 | .895 | .749 |
| E2GAN | .757 | .796 | .794 | .796 | .796 | .795 |
| Attn | .272 | .304 | .324 | .347 | .373 | .406 |
| GLIMA-a | .194 | .203 | .213 | .229 | .240 | .261 |
| GLIMA-g | .195 | .199 | .209 | .221 | .234 | .255 |
| GLIMA-l | .215 | .221 | .230 | .241 | .254 | .268 |
| GLIMA | **.186** | **.198** | **.203** | **.213** | **.229** | **.242** |

*5) Attention Visualization:* To analyze the short-term and long-term dependencies captured by our method, we visualize the attention scores obtained by the multi-directional self-attention mechanism. In Figure 2 and Figure 3, we pick two multivariate time series data samples from KDD18 dataset, and visualize the time-level and feature-level attention maps along with their corresponding variable values. The darker color indicates a higher attention score. Note that since we use a mask to remove the effect of an observed variable on its own estimated values, the diagonal units in the attention maps are set to zero.

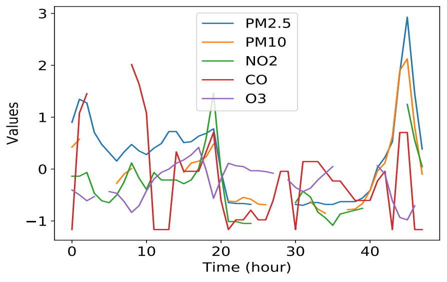
From Figure 2a on temporal attention, we can find that the data points at most timestamps have stronger correlation with nearby timestamps, which indicates that the last few timestamps have more impact on current timestamp for this data sample. From Figure 2b on feature-level attention, we can see that PM2.5, NO2 and CO have higher attention score with each other. This can be verified by variable dynamics as shown in Figure 2c, where the temporal patterns of the three variables are closely related to each other.



(a) Attention across time.



(b) Attention across features.



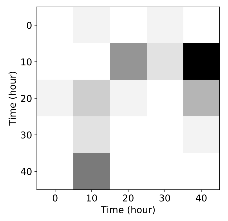
(c) The corresponding time series.

Fig. 2: Visualization of attention scores: case study 1.

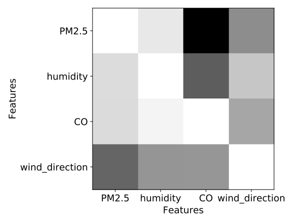
From Figure 3a, we can find that the data points near timestamps 10 and 40 have stronger correlation than with others. This can be verified from the original data as depicted in Figure 3c. Similarly, from Figure 3b and Figure 3c, we can see that the close relationship between PM2.5 and CO has been captured by the attention map.

## V. Conclusion

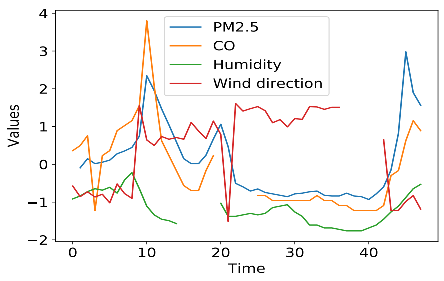
In this paper, we propose GLIMA, a deep imputation model for multivariate time series with missing values. GLIMA combines global recurrent imputation that captures the mixed information from all variables and local recurrent imputation that captures patterns specific to individual variables. Specifically, a time decayed multi-variate GRU based on tensor operation is proposed to efficiently calculate the variable-wise hidden states. Moreover, multi-directional self-attention is employed to exploit both the cross-time and cross-feature correlations in the data. The recurrent imputation process and the attention mechanism allow us to capture the short-term and long-term dependencies at the same time. We conduct experiments on three real-world datasets with different missingness patterns



(a) Attention across time.



(b) Attention across features.



(c) The corresponding time series.

Fig. 3: Visualization of attention scores: case study 2.

in comparison with extensive state-of-the-art methods. The experimental results show that the proposed model is not only able to achieve superior imputation performance, but also able to improve downstream tasks where the input time series datasets are pre-imputed.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported in part by the US National Science Foundation under grants IIS-2008208, IIS-1955151, OAC-1934600 and IIS-1938167. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

[1] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, “Recurrent neural networks for multivariate time series with missing values,” *Scientific reports*, 2018.

[2] Y. Zheng, X. Yi, M. Li, R. Li, Z. Shan, E. Chang, and T. Li, “Forecasting fine-grained air quality based on big data,” in *Proceedings of the 21th ACM SIGKDD*, 2015.

[3] M. Bińkowski, G. Marti, and P. Donnat, “Autoregressive convolutional neural networks for asynchronous time series,” in *ICML*, 2018.

[4] A. T. Hudak, N. L. Crookston, J. S. Evans, D. E. Hall, and M. J. Falkowski, “Nearest neighbor imputation of species-level, plot-scale forest structure attributes from lidar data,” *Remote Sensing of Environment*, pp. 2232–2245, 2008.

[5] M. Morup, D. M. Dunlavy, E. Acar, and T. G. Kolda, “Scalable tensor factorizations with missing data.” Sandia National Laboratories, Tech. Rep., 2010.

[6] S. v. Buuren and K. Groothuis-Oudshoorn, “MICE: Multivariate imputation by chained equations in R,” *Journal of statistical software*, pp. 1–68, 2010.

[7] C. F. Ansley and R. Kohn, “On the estimation of arima models with missing values,” pp. 9–37, 1984.

[8] J. Yoon, W. R. Zame, and M. van der Schaar, “Multi-directional recurrent neural networks: A novel method for estimating missing data,” in *International Conference on Machine Learning (ICML) Time Series Workshop*, 2017.

[9] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, “BRITS: Bidirectional recurrent imputation for time series,” in *Advances in Neural Information Processing Systems*, 2018, pp. 6775–6785.

[10] L. Shen, Q. Ma, and S. Li, “End-to-end time series imputation via residual short paths,” in *Asian Conference on Machine Learning*, 2018, pp. 248–263.

[11] H. Yuan, G. Xu, Z. Yao, J. Jia, and Y. Zhang, “Imputation of missing data in time series for air pollutants using long short-term memory recurrent neural networks,” in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, 2018, pp. 1293–1300.

[12] Q. Suo, L. Yao, G. Xun, J. Sun, and A. Zhang, “Recurrent imputation for multivariate time series with missing values,” in *2019 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2019, pp. 1–3.

[13] Y. Luo, X. Cai, Y. Zhang, J. Xu, and X. Yuan, “Multivariate time series imputation with generative adversarial networks,” in *Advances in Neural Information Processing Systems*, 2018.

[14] Y. Luo, Y. Zhang, X. Cai, and X. Yuan, “E2GAN: end-to-end generative adversarial network for multivariate time series imputation,” in *International Joint Conference on Artificial Intelligence*, 2019, pp. 3094–3100.

[15] Y. Luo, P. Szolovits, A. S. Dighe, and J. M. Baron, “3D-MICE: integration of cross-sectional and longitudinal imputation for multi-analyte longitudinal clinical data,” *Journal of the American Medical Informatics Association*, pp. 645–653, 2017.

[16] H.-F. Yu, N. Rao, and I. S. Dhillon, “Temporal regularized matrix factorization for high-dimensional time series prediction,” in *Advances in neural information processing systems*, 2016.

[17] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[18] Y.-J. Kim and M. Chi, “Temporal belief memory: Imputing missing data during rnn training.” in *In Proceedings of International Joint Conferences on Artificial Intelligence*, 2018.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[21] J. Ma, Z. Shou, A. Zareian, H. Mansour, A. Vetro, and S.-F. Chang, “CDSA: Cross-dimensional self-attention for multivariate, geo-tagged time series imputation,” *arXiv preprint arXiv:1905.09904*, 2019.

[22] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, “Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting,” in *Advances in Neural Information Processing Systems*, 2019.

[23] S. Huang, D. Wang, X. Wu, and A. Tang, “DSANet: Dual self-attention network for multivariate time series forecasting,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2129–2132.

[24] R. Sen, H.-F. Yu, and I. S. Dhillon, “Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting,” in *Advances in Neural Information Processing Systems*, 2019, pp. 4838–4847.

[25] X. Tang, H. Yao, Y. Sun, C. C. Aggarwal, P. Mitra, and S. Wang, “Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values.” in *AAAI*, 2020, pp. 5956–5963.

[26] L. Vincent and N. Thome, “Shape and time distortion loss for training deep time series forecasting models,” in *Advances in Neural Information Processing Systems*, 2019, pp. 4189–4201.

[27] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang, “Multi-horizon time series forecasting with temporal attention learning,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2527–2535.

[28] Y. Matsubara and Y. Sakurai, “Dynamic modeling and forecasting of time-evolving data streams,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 458–468.

[29] P. Deshpande and S. Sarawagi, “Streaming adaptation of deep forecasting models using adaptive recurrent units,” in *Proceedings of the 25th ACM SIGKDD*, 2019.

[30] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, “Deep state space models for time series forecasting,” in *Advances in neural information processing systems*, 2018, pp. 7785–7794.

[31] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, “Deepar: Probabilistic forecasting with autoregressive recurrent networks,” *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[32] T. Guo, T. Lin, and N. Antulov-Fantulin, “Exploring interpretable LSTM neural networks over multi-variable data,” in *ICML*, 2019.

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of CVPR*, 2017, pp. 4700–4708.

[34] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” 2015.

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

[36] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[37] X. Yi, Y. Zheng, J. Zhang, and T. Li, “ST-MVL: Filling missing values in geo-sensory time series data shenzhen institutes of advanced technology, chinese academy of sciences,” in *Proceedings of International Joint Conferences on Artificial Intelligence*, 2016, pp. 9–15.

[38] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[39] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[40] A. Liaw, M. Wiener *et al.*, “Classification and regression by randomforest,” *R news*, vol. 2, no. 3, pp. 18–22, 2002.