ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2024, Vol. 45, No. 11, pp. 5948–5966. © Pleiades Publishing, Ltd., 2024.

SANNI: Online Imputation of Missing Values in Multivariate Time Series Based on Deep Learning and Behavioral Patterns

A. A. Yurtin[1]* and M. L. Zymbler[1]**

(Submitted by A. M. Elizarov)

[1] *South Ural State University, Chelyabinsk, 454080 Russia*

Received September 16, 2024; revised September 26, 2024; accepted October 10, 2024

Abstract—Currently, in a wide spectrum of applications, to avoid the processing and analysis of incomplete time series, end-users need efficient and accurate approaches to online imputation of missing values. In the article, we introduce a novel method called SANNI (snippet and artificial neural network-based imputation) for the recovery of missing values in multivariate time series coming online. SANNI leverages behavioral patterns (called snippets) that are subsequences representing an actor's typical activities, which are reflected by the time series. Preprocessing is performed for each series of a representative fragment of the input data, where we normalize all the subsequences with non-`NaN` values and discover snippets. To impute, our method employs two deep learning models: Recognizer and Reconstructor. Given a multivariate subsequence ended by a missing value, Recognizer outputs a snippet to which the subsequence of the series is the most similar. Reconstructor, for each series, imputes missing values using the snippet discovered previously and results taken from Recognizer. In the extensive experiments, SANNI on average outperforms state-of-the-art competitors over time series from diverse subject domains related to an actor with predefined activities as well as under the blackout scenario.

DOI: `10.1134/S1995080224606854`

Keywords and phrases: *time series, missing values, imputation, deep learning, convolutional neural network, recurrent neural network, gated recurrent unit, snippet, MPdist*

INTRODUCTION

Currently, there is a wide spectrum of applications, where time series need to be efficiently processed online: Internet of Things [1], digital twins [2], smart control [3], and so on. In such domains, software and hardware failures or human factors lead to missing values, which should be imputed as long as they are encountered to avoid the processing and analysis of incomplete time series. Thus, the development of efficient and accurate approaches to online imputation of missing values in time series is a topical issue [4]. At the moment, in reply to the challenge above, numerous both unsupervised approaches [4, 5] and deep learning models [6, 7] are proposed.

In this study, we address the problem of online imputation of multivariate time series tackling the following challenges. First, we treat time series imputation as an actor activity prediction [8] task in the case when data with activity labels do not exist beforehand. Second, among the different scenarios of multiple incomplete series, we focus on the most severe one, blackout [5], where all sensors go quiet simultaneously, causing widespread and aligned missing blocks. In summary, the main contributions of this article are as follows:

*E-mail: `iurtinaa@susu.ru`

**E-mail: `mzym@susu.ru`

- We introduce a novel method called SANNI (Snippet and Artificial Neural Network-based Imputation) for recovery of missing values in multivariate time series coming online. SANNI exploits behavioral patterns (called snippets [9]), which are discovered as a part of data preprocessing. Our method employs two deep learning models, Recognizer and Reconstructor, that are based on recurrent neural networks and together impute missing values. Recognizer takes a current multivariate subsequence ended by a missing value, and for each series, it outputs a snippet to which the current subsequence of the series is the most similar. Reconstructor, for each series, imputes missing values using the results taken from Recognizer.

- We carry out extensive experiments to evaluate our method against state-of-the-art both unsupervised approaches and deep learning models over time series from diverse subject domains for various scenarios. In the experiments, on average, SANNI outperforms the rivals in terms of accuracy for both the target data category and imputation scenario, namely, time series from subject domains related to an actor with several predefined activities and blackout, respectively. In addition, we establish a repository [10], which contains the source code, data, plots, etc. to facilitate the reproducibility of our study.

The remainder of the article is organized as follows. In Section 1, we briefly discuss related works. Section 2 describes the notation and formal definitions our approach is based on. Section 3 introduces the method for imputation of missing values in multivariate time series. In Section 4, we discuss the results of the experimental evaluation of our method. Finally, in Conclusions, we summarize the results obtained and suggest directions for further research.

1. RELATED WORK

Currently, in subject domains related to time series processing, the development of accurate, efficient, and parameterizable approaches to the imputation of missing data blocks remains a topical issue [5]. At the moment, the research community has proposed a wide spectrum of both unsupervised approaches and deep learning models for the imputation of missing values in time series. The following unsupervised algorithms are worth noting as state-of-the-art [4, 5]: CDRec [11], DynaMMo [12], ORBITS [4], ROSL [13], GROUSE [14], SoftImpute [15], SVDImpute [16], SVT [17], and TeNMF [18]. Modern deep learning-based imputation methods employ a wide range of neural network architectures [6, 7]: generative-adversarial networks (e.g., E^2GAN [19], BRNN-GAN [20]), transformers (e.g., SAITS [21], STING [22]), autoencoders (e.g., NAOMI [23], GP-VAE [24]), recurrent neural networks (e.g., BRITS [25], M-RNN [26]), and so on. In our brief overview of related works, we consider only the above-mentioned BRITS and M-RNN (yet do not avoid comparison with all approaches in the experiments) since they are typical representatives of approaches based on recurrent neural networks, which are closest to our study.

The BRITS (Bidirectional Recurrent Imputation for Time Series) model [25] imputes multidimensional time series through two layers consisting of recurrent neurons. The first and second layers process, respectively, the input subsequence and its copy, where the points are taken in reverse order. In each above layer, the number of neurons equals to the input subsequence length, and each neuron predicts the subsequence's next point, taking into account all the preceding points. If the ith point is missed, then it is imputed as the average of the predictions of the $(i-1)$th points from both layers above. Then, the ith point serves as an input of the $(i+1)$th neuron. For the learning, BRITS prescribes to randomly insert "synthetic" missing values in addition to "natural-born" ones, and involves subsequences with missing values, where at least one of them is "synthetic". Learning over a subsequence, BRITS predicts all its points (i.e., non-`NaN` and `NaN` ones). During the learning, the loss function is calculated only over the "synthetic" missing values. However, in the end, BRITS is able to achieve high imputation accuracy since accumulation of error due to the prediction of "natural-born" missing values significantly affects the calculation of the loss function. We can point out two following limitations of BRITS. First, since such an approach employs RNN (Recurrent Neural Network) neurons, not GRU (Gated Recurrent Units) [27] or LSTM (Long Short-term Memory) [28] ones, it may cause the vanishing gradient problem [29]. Second, for long time series, the accuracy of BRITS deteriorates since it does not take into account long-period dependencies both inside and across the series.

The M-RNN (Multi-directional Recurrent Neural Network) model [26] employs two blocks of neurons, which are trained simultaneously using the same predefined subsequence length. The first block is based on the Bi-RNN (Bi-directional Recurrent Neural Network) architecture [30] and performs interpolation of missing values in each series separately from the other ones. The second block employs fully connected layers and imputes each interpolated multivariate point above through extraction of the dependencies across the series. Since the interpolation block processes only missing points, not the subsequences containing them, it leads to small-sized models. However, such an approach suffers from relatively low accuracy since it does not take into account long-period dependencies across series.

2. PRELIMINARIES

Prior to detailing the proposed approach for imputation of missing values in multivariate time series, below, in Sections 2.1, 2.2, and 2.3, respectively, we introduce basic notation and brief description of the snippet concept [9] and the MPdist distance measure [31] our development is based on.

2.1. Basic Notation

A *univariate time series* is a chronologically ordered sequence of real-valued numbers:

$$T = \{t_i\}_{i=1}^{n}, \quad t_i \in \mathbb{R}. \tag{1}$$

The length of a time series, n, is denoted by $|T|$.

A *subsequence* $T_{i,m}$ of a univariate time series T is its subset of m successive elements that starts at the ith position

$$T_{i,m} = \{t_k\}_{k=i}^{i+m-1}, \quad 1 \le i \le n-m+1, \quad 3 \le m \ll n. \tag{2}$$

A *multivariate time series* is an ordered sequence of several equal-length univariate time series that are semantically associated and synchronized by time with each other. Let us denote a dimension of the multivariate time series as the positive integer d ($d > 1$). Similarly to the univariate case, we denote a multivariate time series, its subsequence, and its point as $\boldsymbol{T}$, $\boldsymbol{T}_{i,m}$, and $\boldsymbol{t}_i$, respectively, and define them as below:

$$\boldsymbol{T} = [\{T^{(k)}\}_{k=1}^{d}]^\intercal, \tag{3}$$

$$\boldsymbol{T}_{i,m} = [\{T_{i,m}^{(k)}\}_{k=1}^{d}]^\intercal, \tag{4}$$

$$\boldsymbol{t}_i = [\{t_i^{(k)}\}_{k=1}^{d}]^\intercal. \tag{5}$$

Hereinafter, `NaN` denotes a missed value.

2.2. Snippets

Informally speaking, for the given univariate time series that measures some actor's behavior, its *snippets* are subsequences that represent typical activities of the actor. The snippet concept and the Snippet-Finder algorithm to discover time series snippets are proposed in [9]. In our study, we employ the PSF (Parallel Snippet-Finder) algorithm [32] that accelerates snippet discovery on GPU. Snippets are formally defined as follows.

For the given subsequence length m, let us split the given time series T into a set of *segments*, i.e., non-overlapping subsequences of T. Since $m \ll n$, then without loss of generality, we suppose that n is a multiple of m, and the set of segments, S_T^m is defined as below

$$S_T^m = \{S_i\}_{i=1}^{n/m}, \quad S_i = T_{m\cdot(i-1)+1,\,m}. \tag{6}$$

Snippets of T are to be selected from S_T^m, and we introduce the positive integer K ($1 \le K \le n/m$), the number of snippets (i.e., typical activities of the actor above) we are interested in. Next, we define the set of m-length snippets C_T^m as below:

$$C_T^m = \{C_i\}_{i=1}^{K}, \quad C_i \in S_T^m. \tag{7}$$

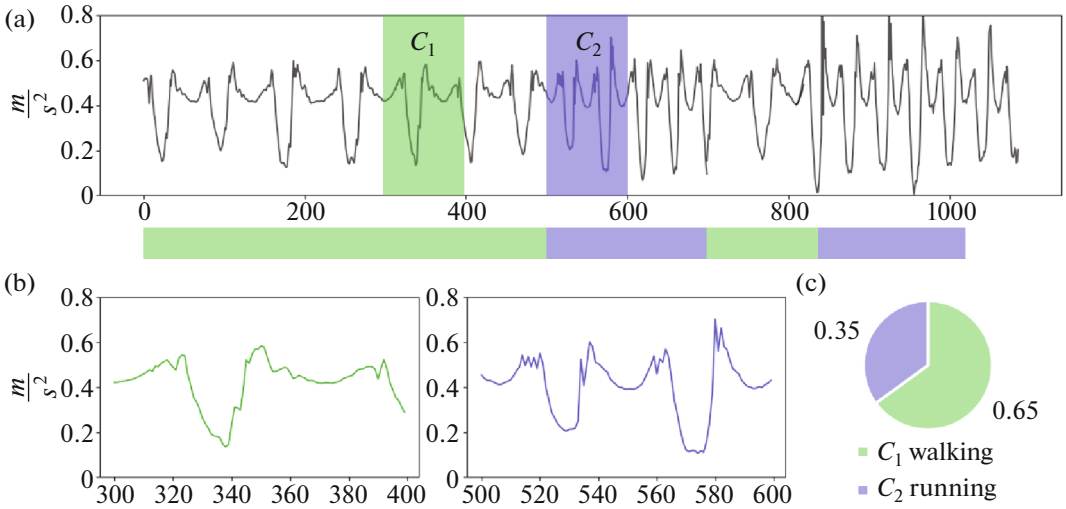


Fig. 1. An example of a univariate time series and its snippets. (a) accelerometer data collected during an individual's walking and running; (b) two-color streak showing the time series labeling as a result of the snippet discovery; (c) snippets discovered and a diagram of their fraction.

A snippet is associated with the following attributes: an index, a set of nearest neighbors, and a fraction. For the given snippet $C_i \in C_T^m$, we denote the above attributes as $C_i.index$, $C_i.NN$, and $C_i.frac$, respectively.

The snippet's index is a number of the segment that corresponds to the snippet:

$$C_i.index = j \Leftrightarrow S_j = T_{m\cdot(j-1)+1,\,m}. \tag{8}$$

The snippet's nearest neighbors are a set of subsequences that are the most similar to the corresponding segment

$$C_i.NN = \{T_{j,m} \mid S_{C_i.index} = \arg\min_{1\le s\le n/m} \mathrm{MPdist}(T_{j,m}, S_s), 1 \le j \le n-m+1\}, \tag{9}$$

where $\mathrm{MPdist}(\cdot,\cdot)$ is the Euclidean distance-based similarity measure proposed in [31] and briefly described below in Section 2.3.

The snippet's fraction is a ratio of the number of the snippet's nearest neighbors to the total number of m-length subsequences in the time series

$$C_i.frac = \frac{|C_i.NN|}{n-m+1}. \tag{10}$$

Finally, in the C_T^m set, snippets are ordered in descending order of their fraction

$$\forall C_i, C_j \in C_T^m: \quad i < j \Leftrightarrow C_i.frac \ge C_j.frac. \tag{11}$$

Figure 1 illustrates the snippet concept for a time series from the PAMAP2 dataset [33], which is collected by an accelerometer worn on an individual during walking and running.

Finally, for the multivariate time series $\boldsymbol{T}$, let us call the set $\boldsymbol{C}_{\boldsymbol{T}}^m$, which combines snippets across all the series, a *snippet dictionary*

$$\boldsymbol{C}_{\boldsymbol{T}}^m = \bigcup_{k=1}^{d} C_{T^{(k)}}^m. \tag{12}$$

2.3. The MPdist Measure

Informally speaking, two equal-length time series are as close to each other w.r.t. MPdist as many of their smaller equal-length subsequences are close to each other w.r.t. the normalized Euclidean distance. Despite the fact that MPdist does not meet the triangular inequality, it is robust w.r.t. spikes, warping, linear trends, etc. [31].

Let us consider A and B, two m-length time series, and the subsequence length ℓ, where $\lceil 0.3m \rceil \leq \ell \leq \lceil 0.8m \rceil$. MPdist is formally defined as following three-step procedure.

At first, let us calculate the matrix profile for A and B w.r.t. the subsequence length ℓ and denote the result as P_{AB}. The matrix profile concept proposed in [34] and defined as a time series, where its ith element is the distance from the ith subsequence of the first time series to its nearest neighbor in the second time series

$$P_{\mathrm{AB}} = \{\mathrm{Dist}(A_{i,\ell}, B_{j,\ell})\}_{i=1}^{m-\ell+1}, \quad B_{j,\ell} = \arg\min_{1 \leq q \leq n/m} \mathrm{Dist}(A_{i,\ell}, B_{q,\ell}), \tag{13}$$

where $\mathrm{Dist}(\cdot,\cdot)$ is a nonnegative symmetric function. Similarly, the matrix profile P_{BA} is defined as below

$$P_{\mathrm{BA}} = \{\mathrm{Dist}(B_{i,\ell}, A_{j,\ell})\}_{i=1}^{m-\ell+1}, \quad A_{j,\ell} = \arg\min_{1 \leq q \leq n/m} \mathrm{Dist}(B_{i,\ell}, A_{q,\ell}). \tag{14}$$

Secondly, let us concatenate P_{AB} with P_{BA} denoting the resulting time series as P_{ABBA}:

$$P_{\mathrm{ABBA}} = P_{\mathrm{AB}} \bullet P_{\mathrm{BA}}, \quad |P_{\mathrm{ABBA}}| = 2(m - \ell + 1), \tag{15}$$

where the symbol $\bullet$ denotes a concatenation of two operands.

Finally, let us sort the elements of P_{ABBA} in ascending order and denote the result as $sorted P_{\mathrm{ABBA}}$. To calculate MPdist between A and B w.r.t. the subsequence length ℓ, we choose the kth element of $sorted P_{\mathrm{ABBA}}$, where k is a predefined parameter with typical value $k = \lceil 0.1m \rceil$, or the maximal element of P_{ABBA} if the subsequence length ℓ is close to the time series length m:

$$\mathrm{MPdist}_{\ell}(A, B) = \begin{cases} sorted P_{\mathrm{ABBA}}(k), & |P_{\mathrm{ABBA}}| > k \\ sorted P_{\mathrm{ABBA}}\big(2(m - \ell + 1)\big), & \text{otherwise.} \end{cases} \tag{16}$$

In equations (13) and (14), as the $\mathrm{Dist}(\cdot,\cdot)$ function, we use the z-normalized Euclidean distance that is defined as below

$$\mathrm{ED}_{\mathrm{norm}}(X, Y) = \mathrm{ED}(\hat{X}, \hat{Y}) = \sqrt{\sum_{i=1}^{\ell} (\hat{x}_i - \hat{y}_i)^2},$$

$$\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x}, \quad \mu_x = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i, \quad \sigma_x = \sqrt{\frac{1}{\ell} \sum_{i=1}^{\ell} x_i^2 - \mu_x^2}. \tag{17}$$

3. METHOD

At this point, we are ready to introduce our method for imputation of missing values in multivariate streaming time series called SANNI (snippet and artificial neural network-based imputation). Below, Section 3.1 presents the general architecture of our approach. In Section 3.2, we outline the data preprocessing phase. Sections 3.3 and 3.4 describe Recognizer and Reconstructor, respectively, two neural network models, which are included in our method.

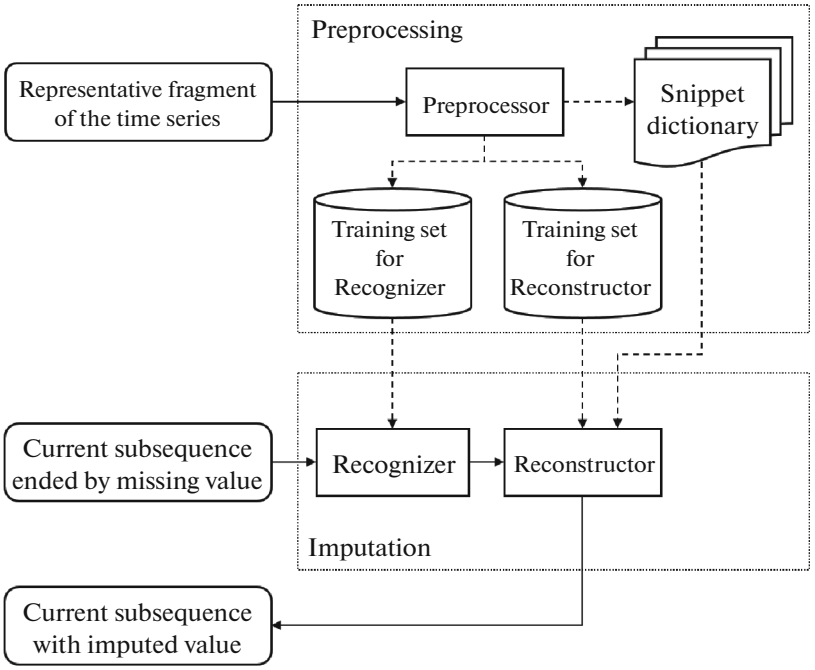


Fig. 2. General architecture of SANNI.

3.1. General Architecture

Figure 2 depicts the general architecture of SANNI. For processing, we take a previously stored fragment of the time series and make the following assumptions about the data. We assume that such a fragment contains measurements of all the actor's basic activities that the time series represents. In addition, we assume that a domain expert predefines a meaningful subsequence length (i.e., snippet length). Moreover, for each series in the fragment, the number of subsequences with at least one `NaN` value does not exceed 50 percent of the total number of subsequences.

In short, SANNI performs as follows. For each series in the fragment, Preprocessor normalizes all the subsequences with non-`NaN` values, discovers their snippets, and creates the snippet dictionary. Next, Preprocessor builds training sets for two deep learning models, Recognizer and Reconstructor, that together impute missing values. Further, imputation is performed as follows. Recognizer takes a current multivariate subsequence ended by a missing value, and for each series, it outputs a snippet to which the current subsequence of the series is the most similar. Finally, for each series, Reconstructor imputes missing values using the snippet dictionary and results taken from Recognizer.

3.2. Data Preprocessing

Hereinafter in this section, as $\boldsymbol{T}$, we denote the above-mentioned fragment of the input multivariate time series. For each series $T^{(k)}$ of $\boldsymbol{T}$, to bring each point t_i of the series (excluding `NaN`s) to the range $[0, 1]$, we perform min-max normalization as below

$$\hat{t}_i^{(k)} = \frac{t_i^{(k)} - \min\limits_{1\le j\le n} t_j^{(k)}}{\max\limits_{1\le j\le n} t_j^{(k)} - \min\limits_{1\le j\le n} t_j^{(k)}}, \quad 1 \le k \le d. \tag{18}$$

Next, we exclude from processing each subsequence $\boldsymbol{T}_{i,m}$, which contains at least one `NaN`. Then, for each series $T^{(k)}$ of $\boldsymbol{T}$, we discover snippets by the PSF algorithm [32] and combine the results into the snippet dictionary, $\boldsymbol{C}_{\boldsymbol{T}}^{m}$.

Further, we denote a training set as $D = \{\langle \boldsymbol{X}, \boldsymbol{Y}\rangle\}$, where X and Y correspond to the input and output attributes of the above tuple, respectively.

As for the tuple of the Recognizer's training set, as an input attribute, we take a multivariate subsequence without NaNs in any series, where we exclude the last multivariate point. As an output attribute, we take a column vector of integers, where its element corresponds to the number of the snippet, which meets the following property: in the respective series, the univariate subsequence is among the nearest neighbors of the above snippet. Formally speaking, $D_{\text{Recognizer}}$ is defined as below

$$D_{\text{Recognizer}} = \{\langle \boldsymbol{X}, \boldsymbol{Y}\rangle \mid X^{(k)} = T^{(k)}_{i,\,m-1},\ t^{(k)}_i \neq \texttt{NaN},$$
$$Y^{(k)} = s,\ T^{(k)}_{i,\,m} \in C^{(k)}_s.NN,\ 1 \leq i \leq n-m+1,\ 1 \leq k \leq d,\ 1 \leq s \leq K\}. \tag{19}$$

The Reconstructor's input attribute of the training set tuple is formed similar to the Recognizer's one; it is a multivariate subsequence without NaN values in any series, where the last multivariate point is changed to the special NIL point, which is simply a column vector of -1 values. The excluded multivariate point is assumed as an output attribute. Thus, we define $D_{\text{Reconstructor}}$ as below

$$D_{\text{Reconstructor}} = \{\langle \boldsymbol{X}, \boldsymbol{Y}\rangle \mid X^{(k)} = T^{(k)}_{i,\,m-1} \bullet \texttt{NIL},\ \texttt{NIL} = -1,$$
$$Y^{(k)} = t^{(k)}_{i+m},\ 1 \leq i \leq n-m+1,\ 1 \leq k \leq d\}. \tag{20}$$

3.3. Recognizer

Figure 3 depicts the structure of Recognizer. The model consists of the following layers applied one after another: three convolutional layers, one recurrent layer, and two fully connected layers. Each convolutional layer provides 256 channels with the kernel size 5 to extract features from the input subsequence. After each convolutional layer, we place the Max-pooling node with the window size 2 to create down-sampled feature maps and employ the ReLU activation function. The recurrent layer includes GRUs (Gated Recurrent Units), each with an $m/4$-sized hidden state, to analyze the features extracted by the previous layers, taking into account the chronology of features. This layer employs Leaky ReLU as an activation function to avoid the dying ReLU problem [35].

The next two fully-connected layers are in charge of the final evaluation of the input subsequence, where the former and the latter layers consist of, respectively, $32 \cdot m$ and $d \cdot K$ neurons, and the latter layer calculates the *membership matrix*. In such a matrix, a row represents the respective series and is a vector of probabilities of the fact that the subsequence is the nearest neighbor of the respective snippet. Let us denote the membership matrix for the given multivariate subsequence as $\boldsymbol{P}_{\boldsymbol{T}_{i,\,m}}$, then its formal definition is as follows

$$\boldsymbol{P}_{\boldsymbol{T}_{i,\,m}} = [\{P_{T^{(k)}_{i,\,m}}\}_{k=1}^{d}]^{\intercal}\colon\ P^{(k)}(j) = \Pr(T^{(k)}_{i,\,m} \in C^{m}_{T^{(k)}}),\quad \sum_{j=1}^{K} P^{(k)}(j) = 1,$$
$$1 \leq i \leq n-m+1,\quad 1 \leq k \leq d,\quad 1 \leq j \leq K. \tag{21}$$

3.4. Reconstructor

In Fig. 4, we show the structure of Reconstructor. Given a multivariate subsequence, where the last NaNs are changed to NILs, Reconstructor imputes each series, where the last point is missed. We build a three-dimensional tensor, which serves as an input for Reconstructor. In the tensor, for each time series dimension, we keep the two-row *layout matrix*, at which the first row is an input subsequence, where we substitute the last missing value by NIL, and the second row is the respective snippet taken from the membership matrix produced by Recognizer. Let us denote the above-mentioned tensor as $\boldsymbol{M}_{\boldsymbol{T}_{i,\,m}}$, then it is formally defined as follows

$$\boldsymbol{M}_{\boldsymbol{T}_{i,\,m}} = \{M_{T^{(k)}_{i,\,m}}\}_{k=1}^{d},\quad M_{T^{(k)}_{i,\,m}} \in \mathbb{R}^{2\times m},$$
$$M_{T^{(k)}_{i,\,m}}(1,\cdot) = T^{(k)}_{i,\,m-1} \bullet \texttt{NIL},\quad \texttt{NIL} = -1,$$
$$M_{T^{(k)}_{i,\,m}}(2,\cdot) = C^{m}_{T^{(k)}}\left(\arg\max_{1\leq j\leq K} P_{T^{(k)}_{i,\,m}}(j)\right),\quad 1 \leq k \leq d. \tag{22}$$

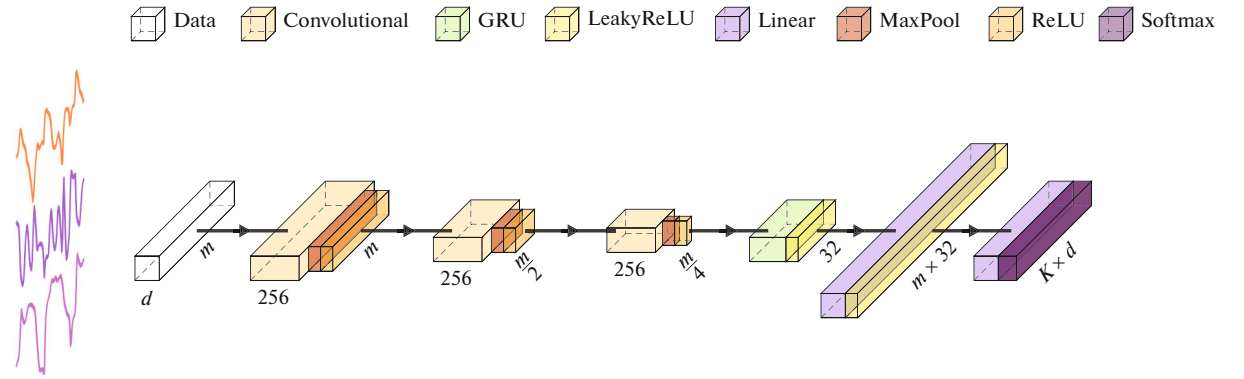


Fig. 3. The structure of Recognizer.

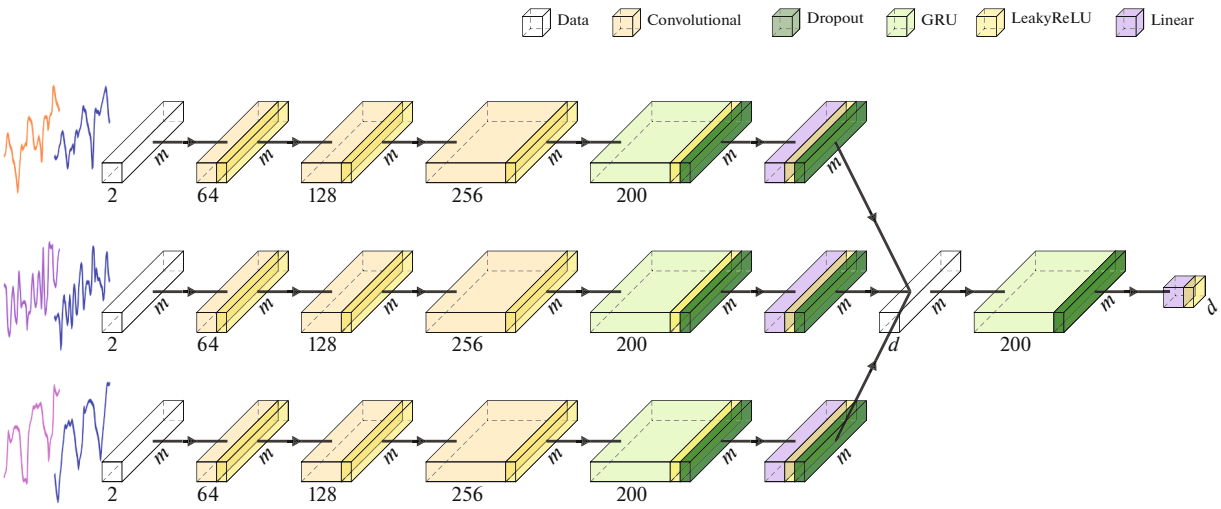


Fig. 4. The structure of Reconstructor.

Then, for each series, Reconstructor processes the respective layout matrix through three convolutional layers (with kernel size 5 and 64, 128, and 256 feature maps, respectively), which extract features related to the similarity of the subsequence and the snippet it is the most similar to. Further, for each series, we place a single GRU with an m-sized hidden state and a linear layer of m neurons, which together discover implicit time-related dependencies between the subsequence and its respective snippet in the previously extracted features. For the two above layers, during the learning, we apply the dropout operation to their neurons with the predefined probability p, where $0 < p < 1$ and the typical value for such a parameter is $p = 0.2$.

Next, we combine outputs from linear layers into the $(d \times m)$-sized matrix and treat it as an input for a single GRU with an m-sized hidden state that discovers implicit time-related dependencies across the series. As before, we apply the dropout operation to the GRU above. Finally, d linear neurons represent the model's output.

To learn Reconstructor, we employ a training set produced by Recognizer since it is more likely contains incorrectly classified samples than in the case when the training set is prepared through the Snippet-Finder [9] or PSF algorithm [32]. By doing so, after the learning, we better adapt Reconstructor for incorrect data produced by Recognizer.

4. EXPERIMENTAL EVALUATION

To evaluate the proposed method, we test SANNI over various real-world and synthetic time series in comparison with state-of-the-art analogs. We designed the experiments to be easily reproducible

Table 1. Datasets employed in the experiments

No.	Dataset	Length, $n \times 10^3$	Dimension, d	Subject domain
Category A: Actor with activities				
1.	Electricity	5	9	Power demand of households in EU
2.	Madrid	25	10	Automatic vehicle registration in Madrid
3.	NREL	8.7	9	Power demand of a research laboratory in USA
4.	PAMAP	50	10	Wearable sensors during an adult's activities
5.	WalkRun	37	11	
Category B: Seasonality and/or cyclicity				
6.	BAFU	50	10	Water discharge in Swiss rivers
7.	Climate	5	10	Weather in North America
8.	MAREL	50	10	Seawater in the English Channel
9.	MeteoSwiss	10	10	Weather in Switzerland
10.	Saaleaue	23	14	Weather in Germany
Category C: Stochasticity				
11.	BTC	2.5	12	Trading characteristics of cryptocurrencies
12.	Soccer	500	10	Wearable sensors during football players' activities

with our repository [10], which contains the source code and all the datasets used in this work. Below, Section 4.1 describes the hardware and time series employed in the experiments, and Section 4.2 presents the experimental results and discussion.

4.1. The Experimental Setup

Datasets. For the experiments, we employed the datasets summarized in Table 1. In datasets, we distinguish three categories of multivariate time series that can informally be described as follows. *Category A* includes time series in such domains, where it is possible to point out one or more actors that together demonstrate several predefined activities. In *Category B*, we collect seasonal and cyclic time series, where fluctuations are of known and non-fixed periods, respectively. Finally, *Category C* represents stochastic time series that reflect an actor's non-deterministic behavior without seasonality or cyclicity. It is worth noting that *Category A* is the target one for our study.

In the experiments, we evaluate SANNI over the following datasets of Category A. The Electricity dataset [36] contains measurements of total electric power consumption during 2011–2014 in several households located in the European Union. The Madrid dataset [37] contains time series from automatic vehicle registration (AVR) devices installed on city roads and motorways in Madrid during 2014–2016. The NREL dataset [38] introduces hourly power demand data of various engineering systems in the National Renewable Energy Laboratory, USA, during 2011. The PAMAP dataset is our excerpt from the well-known PAMAP2 [33] dataset with various physical activity monitoring data on several subjects, where we choose a time series on an individual with six activities. The WalkRun dataset is our collection of the accelerometer, gyroscope, and magnetometer measurements during an individual's alternation of running and walking with wearable sensors. For the Electricity, NREL, and Madrid datasets, respectively, households, staff, and AVR devices exhibit different activities in working days and holidays. For the PAMAP and WalkRun datasets, an actor and its activities are obvious.

Next, Category B comprises the following datasets. The BAFU dataset [39] consists of time series showing water discharge in different Swiss rivers. The Climate dataset [40] presents monthly aggregated

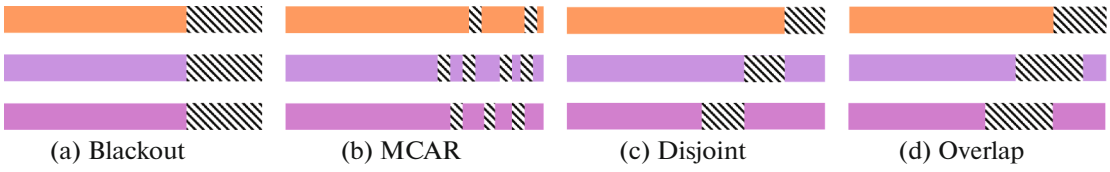


Fig. 5. Scenarios of the experiments (the missing values are represented by hatching).

climate data collected from weather stations in various locations of North America in 1990–2002. The MAREL dataset [41] presents data on diverse chemical and biological characteristics of seawater in the English Channel. The MeteoSwiss dataset [42] provides various weather measurements in Switzerland: air temperature, precipitation, wind, etc. The Saaleaue dataset [43] is gathered in 2009–2016 by the meteostation at the Max Planck Institute for Biogeochemistry, Germany, and includes various weather measurements: air temperature, humidity, CO_2 concentration, etc.

Finally, the following datasets represent Category C. The BTC dataset [44] provides historical data on Bitcoin, Ethereum, and Monero cryptocurrencies, including the opening price, high price, low price, and closing price. The Soccer dataset [45] is collected during a football match, where sensors are located near the players' boots. This dataset is the longest since the sensors of 200 Hz frequency generate 15K position events per second.

Competitors. In the experiments, we compare SANNI with the following state-of-the-art deep learning methods: NAOMI [23], BRITS [25], GP-VAE [24], M-RNN [26], SAITS [21], and Transformer [21], where we exploit their original implementations. In addition, we involve the following state-of-the-art unsupervised time series imputation algorithms: CDRec [11], DynaMMo [12], ORBITS [4], ROSL [13], GROUSE [14], SoftImpute [15], SVDImpute [16], SVT [17], and TeNMF [18], which are implemented in the ORBITS framework [4].

For each above-mentioned deep learning or unsupervised rival, we set its hyperparameters as recommended, respectively, by the authors of the approach, or in the ORBITS framework [4], to provide the highest possible imputation accuracy. At the same time, to provide a fair comparison, in the experiments, we set the domain-dependent parameter, the subsequence length, to be the same for all deep learning approaches.

Scenarios. In the experiments, we employ four scenarios proposed in [5] when several series have missing blocks: Blackout, MCAR, Disjoint, and Overlap (see Fig. 5). In the Blackout scenario, 100 points are missed at the end of each series. The Blackout scenario poses an accuracy challenge [5] and is the target one for our study. The MCAR (Missing Completely at Random) scenario labels random 10-point blocks in a randomly chosen series until 10 percent of the input time series are marked as missing. For all competitors, we use a random number generator with a fixed seed to ensure identical conditions for the experiment. In the Disjoint scenario, for each series, $\lceil n/d \rceil$-length missing block is established, where for ith series the block starts from the $i\lceil n/d \rceil$th position. The Overlap scenario is similar to Disjoint with the difference that the missing block length is two times longer, and its starting position is two times larger. It is worth noting that some imputation algorithms, namely, GROUSE, ORBITS, SoftImpute, SVDImpute, SVT, and TeNMF, cannot be evaluated under the Blackout scenario, since they require at least one non-`NaN` value in a missed multivariate point.

In the experiments, the missing data obtained according to the above scenarios represent the test set. To establish the training set and validation set, we take, respectively, 75 and 25 percent of the multivariate subsequences chosen randomly in the input time series (including those ones with `NaN` values). As before, for all deep learning competitors, we use a random number generator with a fixed seed to ensure identical conditions for the experiment.

Metrics. To evaluate the imputation accuracy, we use RMSE (Root Mean Square Error) since it is the most commonly used measure in this field [8], which is defined as below

$$\mathrm{RMSE} = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(t_i - \tilde{t}_i)^2}, \tag{23}$$

where t_i is a real point to be imputed, $\tilde{t}_i$ is a point synthesized for imputation, and h is a number of imputed points.

Table 2. Hardware platform of the experiments

Specifications	CPU	GPU
Brand and product line	Intel Xeon	NVIDIA Ampere
Model	E5-2687W v2	A100
Number of cores	8	6 912
Core frequency, GHz	3.40	1.41
Memory, Gb	16	80
Peak performance (double precision), TFLOPS	0.157	9.7

Table 3. Hyperparameters of SANNI

Hyperparameter	Value
Snippet length, m	200
Number of activities, K	2
Optimizer	Adam
Initial learning rate	0.0005
Batch size	128
No. of epochs for Recognizer	100
No. of epochs for Reconstructor	1000
Dropout factor, p	0.2

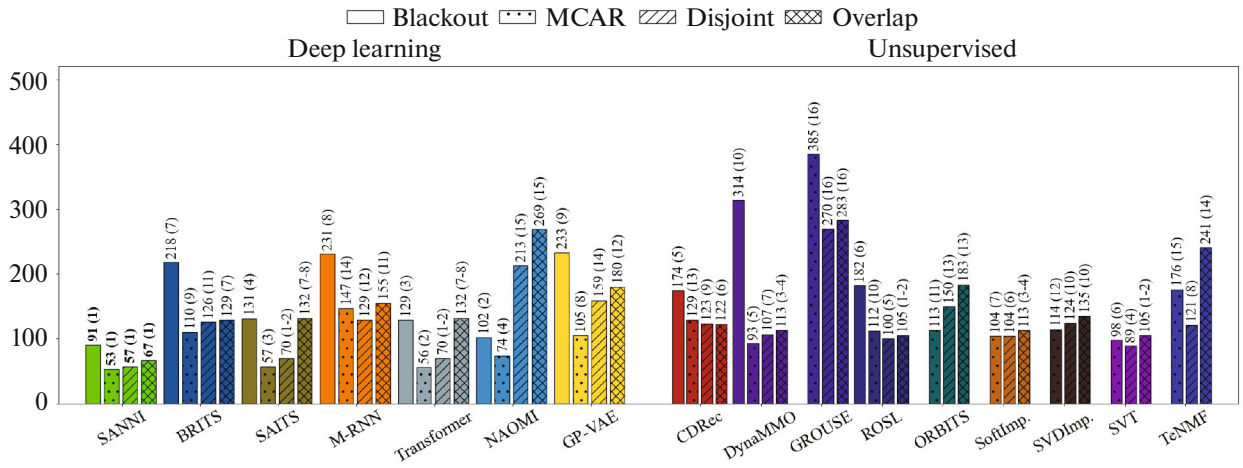
In addition, for each deep learning method involved in the experiments, we measured its performance in terms of running time spent on both learning and imputation. We exclude unsupervised approaches from the comparison because, obviously, in terms of performance, they are significantly ahead of deep learning methods since they do not require learning. We used the Overlap scenario because it has the most missing points and could show the big picture of things without losing generality. Finally, we evaluated the performance of rivals over just the Madrid dataset, since its cardinality is the closest to the average among all the datasets involved in the experiments.

Hardware and hyperparameters. We conducted our experiments using the hardware of the HPC center of the South Ural State University [46], summarized in Table 2.

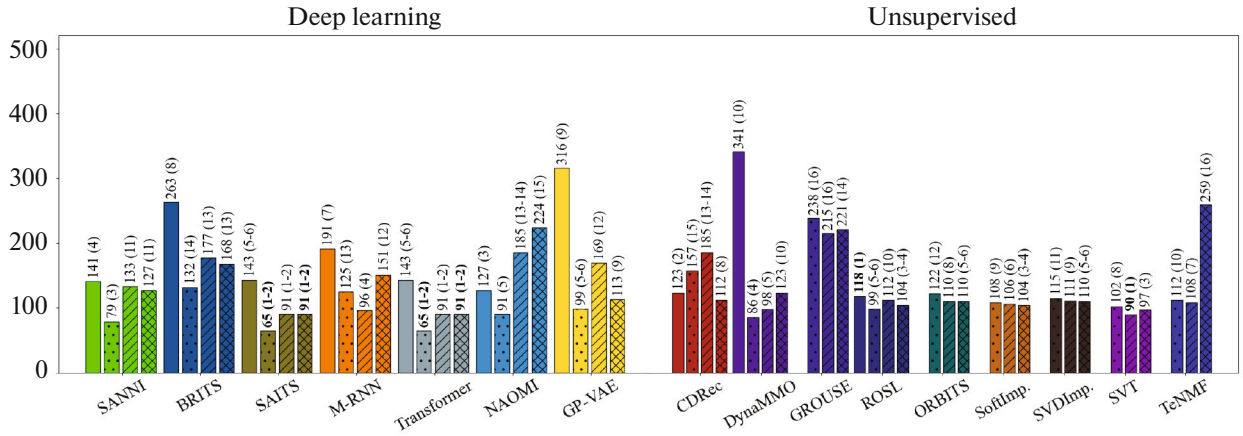
In Table 3, we show hyperparameters of our experiments. Let us remind that in the evaluation, for each deep learning competitor, we exploit the same subsequence length equals to 200.

4.2. Results and Discussion

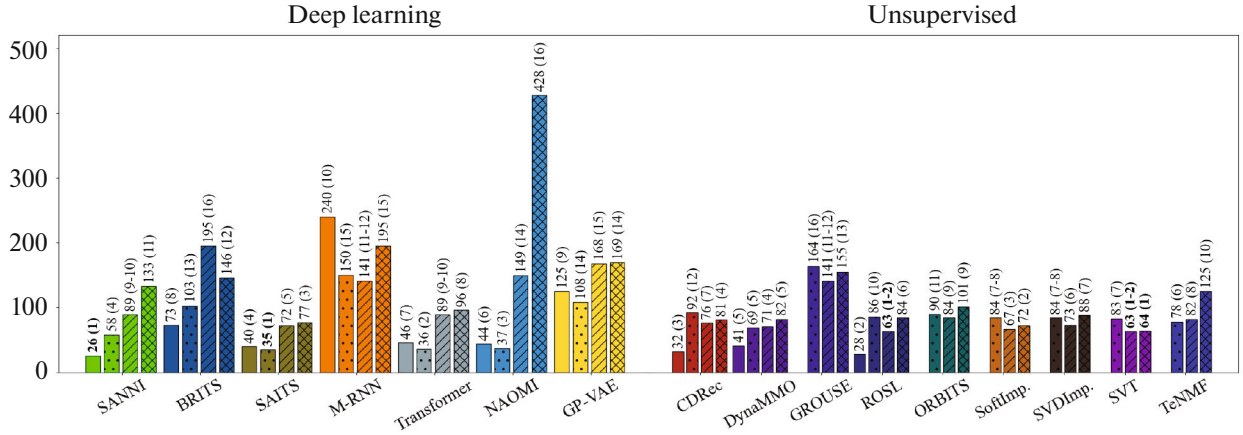
Accuracy. In Fig. 6, we show average imputation accuracy among all the datasets involved in the experiments under all the scenarios grouped by the dataset category (the complete tabular massive of results is presented in Appendix). As can be seen, our approach outperforms all the rivals over time series in the target category (an actor with some predefined activities) under all the scenarios. Over the seasonal/cyclic time series, SANNI relatively succeeds under the Blackout and MCAR scenarios (where it is in the top-4 and top-3 positions, respectively), whereas under the Disjoint and Overlap scenarios it is near the bottom of the rank list. As for the stochastic time series, we can observe a similar picture: our approach performs its best under the Blackout scenario, keeps the top-4 position under MCAR, and is near the bottom under the rest scenarios. In Fig. 7, we illustrate the results above showing 100-point excerpts of ground truth series and imputed ones performed by the top-5 approaches in terms of accuracy over the datasets Madrid, BAFU, and BTC (which represent, respectively, the A,



(a) Category A: Actor with activities



(b) Category B: Seasonality and/or cyclicity



(c) Category C: Stochasticity

Fig. 6. Average imputation accuracy among all the datasets involved under all the scenarios grouped by the dataset category, RMSE $\times 10^{-3}$ and average rank (lower values are better).

B, and C categories) under all the scenarios. Summing up, we conclude that SANNI succeeds both in the target category of time series and under the target imputation scenario.

Performance. Figure 8 depicts the experimental results of the performance of deep learning competitors. As for performance of learning, it can be seen that SAITS and Transformer are far ahead of the rest of their rivals. The reasons are that the above models, first, do not employ recurrent layers, and

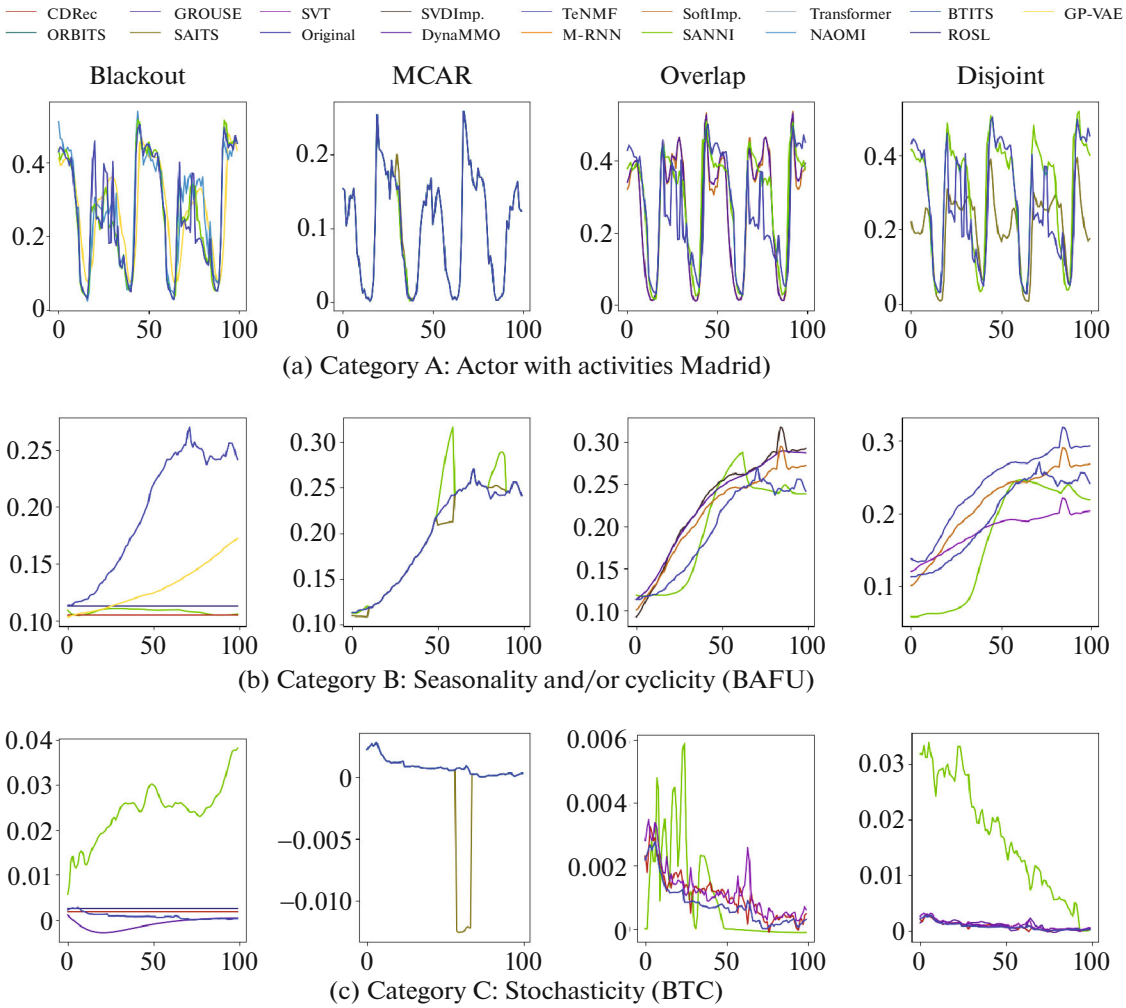


Fig. 7. Examples of imputation under all the scenarios grouped by the dataset category (100-point excerpts of top-5 approaches in terms of accuracy).

second, exploit the self-attention [47] mechanism. SANNI demonstrates a pretty modest performance in learning since, as opposed to its rivals, it includes the overheads on snippet discovery and learning the Recognizer model. Further, GP-VAE, which does not employ recurrent layers as above, demonstrates the best performance on imputation of both a single and all points. SANNI outruns just the rest of the recurrent models, BRITS and M-RNN, being inferior to the rest of their rivals. Thus, our approach would not be only of theoretical interest if it were suitable for imputation of time series coming online.

Let us confirm that our approach fits online imputation. In Building Automation (BA) and Process Automation (PA), the typical update rate of sensors is 10 s and 100 ms, respectively [48]. BA includes diverse control operations applied within buildings: fire control, lighting, heating, water supply, air-conditioner, etc. [49]. PA is common in chemical, pharmaceuticals, mining, oil and gas, metallurgic processes, etc. [50]. Since SANNI imputes one point in 38 ms (see Fig. 8), obviously, our approach can be used to impute sensor data in the above broad classes of subject domains.

CONCLUSIONS

In the article, we addressed a topical problem of the imputation of missing values in time series. We introduced a novel deep learning method called SANNI to recover multivariate time series coming online. Our method leverages behavioral patterns (called snippets), which represent an actor's typical activities, the time series undergoing processing exhibit. A snippet [9] is the given-length subsequence, which is similar to many other subsequences w.r.t. MPdist [31], a bespoke distance measure that is based on the normalized Euclidean metric.

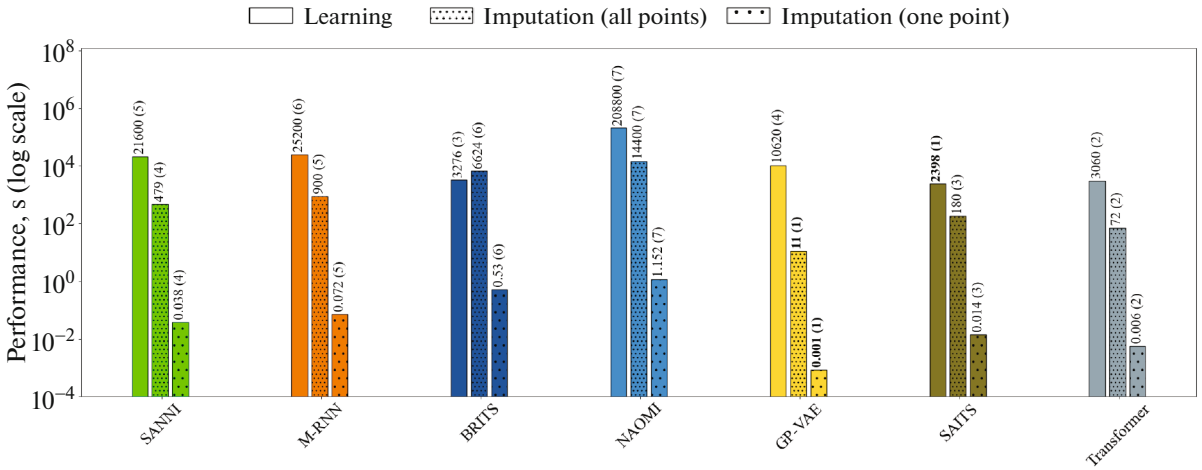


Fig. 8. Performance of deep learning approaches.

SANNI prescribes to predefine a meaningful snippet length and store a representative time series fragment, which includes measurements of all the actor's basic activities that the time series represents. As a preprocessing step, for each series of the fragment above, we normalize all the subsequences with non-`NaN` values, discover snippets, form the snippet dictionary, and build training sets for the two deep learning models that make up the essence of our method: Recognizer and Reconstructor.

Recognizer takes a current multivariate subsequence ended by a missing value, and for each series, it outputs a snippet to which the current subsequence of the series is the most similar. The model consists of the following layers applied one after another: three convolutional layers, one recurrent layer, and two fully connected layers. Each convolutional layer extracts features from the input subsequence. After each convolutional layer, we place Max-pooling and employ ReLU as an activation function. The recurrent layer includes GRUs (Gated Recurrent Units) to analyze the features extracted by the previous layers. This layer employs the Leaky ReLU activation function. The two next fully-connected layers calculate the membership matrix, where a row represents the respective series and is a vector of probabilities of the fact that the subsequence is the nearest neighbor of the respective snippet.

Reconstructor, for each series, imputes missing values using the snippet dictionary and results taken from Recognizer. As an input, Reconstructor takes a bespoke three-dimensional tensor. In such a tensor, for each time series dimension, we keep the two-row layout matrix, in which the first row is a subsequence undergoing imputation, where we substitute the last missing value by -1, and the second row is the respective snippet taken from the membership matrix produced by Recognizer. Then, for each series, Reconstructor processes the respective layout matrix through three convolutional layers, which extract features related to the similarity of the subsequence and the snippet it is the most similar to. Further, for each series, we place a single GRU and a linear layer, which together discover implicit time-related dependencies between the subsequence and its respective snippet in the previously extracted features. For the two above layers, during the learning, we apply the dropout operation. Next, we combine outputs from linear layers into the matrix, which serves as an input for a single GRU to discover implicit time-related dependencies across the series. As before, we apply the dropout operation to the GRU above. Finally, linear neurons represent the model's output. To learn Reconstructor, we employ a training set produced by Recognizer to better prepare Reconstructor for incorrectly classified samples produced by Recognizer.

To evaluate the proposed method, we test SANNI against state-of-the-art unsupervised algorithms and deep learning models over time series from diverse subject domains for various scenarios. Experimental results on imputation accuracy showed that on average, SANNI outperforms the rivals a) over time series from subject domains related to an actor with several predefined activities, and b) under the blackout imputation scenario, which both are the target aspects of our study. As for the performance of imputation, in terms of average time to impute a single data point, SANNI is not among the best deep learning approaches. Nonetheless, we pointed out subject domains where the performance our approach achieves is enough for online imputation. Finally, to facilitate the reproducibility of our study, we establish a repository [10] that contains the source code, data, plots, etc.

In further research, we plan to elaborate our approach, considering motifs and other time series data mining primitives under various distance measures [51, 52] as behavioral patterns.

APPENDIX

Imputation accuracy over all the datasets involved in the experiments under all the scenarios grouped by the dataset category, RMSE $\times 10^{-3}$ and rank (lower values are better)

Method		Dataset													
		Category A: Actor with activities						*Category B: Seasonality and/or cyclicity*					*Category C: Stochasticity*		
Type	Name	Electricity	Madrid	PAMAP	NREL	WalkRun	*Mean*	BAFU	MAREL	Meteo	Saaleaue	*Mean*	BTC	Soccer	*Mean*
Blackout															
Unsupervised	CDRec	253 (6)	213 (6)	118 (6)	194 (5)	91 (5)	174 (5)	78 (3)	194 (8)	130 (3)	**92 (1)**	124 (2)	4 (1-3)	59 (6)	32 (3)
	DynaMMO	349 (9)	286 (9)	252 (10)	378 (9-10)	305 (10)	314 (10)	230 (10)	332 (10)	299 (8)	502 (9)	341 (10)	4 (1-3)	78 (7)	41 (5)
	ROSL	266 (7)	215 (7)	122 (8)	195 (6)	110 (6)	182 (6)	74 (2)	191 (7)	110 (2)	99 (3)	**118 (1)**	4 (1-3)	53 (4)	28 (2)
Deep learning	BRITS	305 (8)	232 (8)	120 (7)	253 (8)	178 (8)	218 (7)	172 (8)	239 (9)	380 (9)	263 (8)	264 (8)	17 (6)	130 (8)	74 (8)
	GP-VAE	447 (10)	129 (3)	102 (3)	378 (9-10)	112 (7)	234 (9)	**67 (1)**	174 (3)	444 (10)	578 (10)	316 (9)	9 (4)	240 (9)	124 (9)
	M-RNN	231 (5)	294 (10)	145 (9)	241 (7)	245 (9)	231 (8)	184 (9)	171 (2)	227 (7)	182 (7)	191 (7)	155 (10)	324 (10)	240 (10)
	NAOMI	142 (2)	37 (2)	111 (4-5)	**138 (1)**	**84 (1-4)**	102 (2)	155 (7)	**170 (1)**	**87 (1)**	94 (2)	126 (3)	61 (9)	**27 (1)**	44 (6)
	SAITS	169 (4)	164 (4-5)	**96 (1-2)**	144 (3-4)	**84 (1-4)**	131 (4)	99 (5-6)	189 (5-6)	164 (5-6)	120 (4-5)	143 (5-6)	35 (7-8)	45 (3)	40 (4)
	Transformer	156 (3)	164 (4-5)	**96 (1-2)**	144 (3-4)	**84 (1-4)**	129 (3)	99 (5-6)	189 (5-6)	164 (5-6)	120 (4-5)	143 (5-6)	35 (7-8)	57 (5)	46 (7)
	SANNI	**84 (1)**	**34 (1)**	111 (4-5)	142 (2)	**84 (1-4)**	**91 (1)**	88 (4)	179 (4)	147 (4)	151 (6)	141 (4)	10 (5)	42 (2)	**26 (1)**
MCAR															
Unsupervised	CDRec	106 (12)	103 (14)	137 (14)	120 (9)	177 (15)	129 (13)	75 (12)	358 (16)	78 (7-9)	119 (13)	158 (15)	60 (7-8)	125 (12)	92 (12)
	DynaMMO	100 (7)	71 (6)	57 (6)	111 (5)	127 (7)	93 (5)	39 (4)	147 (5)	73 (4)	85 (5)	86 (4)	60 (7-8)	79 (5)	70 (5)
	GROUSE	131 (15)	427 (16)	551 (16)	512 (16)	303 (16)	385 (16)	362 (16)	241 (15)	199 (16)	149 (16)	238 (16)	146 (16)	183 (16)	164 (16)
	ORBITS	105 (10-11)	90 (10)	82 (11)	114 (6-7)	174 (14)	113 (11)	109 (15)	192 (11)	85 (11)	104 (9)	122 (12)	67 (11-13)	113 (10-11)	90 (11)
	ROSL	110 (13)	91 (11)	72 (9-10)	141 (12)	144 (8)	112 (10)	59 (8)	169 (8)	78 (7-9)	88 (7)	98 (6)	76 (14)	96 (6)	86 (10)
	SoftImp.	103 (8)	76 (7)	70 (8)	116 (8)	156 (10)	104 (7)	65 (10)	188 (10)	80 (10)	100 (8)	108 (9)	61 (9)	106 (7)	84 (8)
	SVDImp.	105 (10-11)	80 (8-9)	101 (12)	123 (10)	161 (11)	114 (12)	62 (9)	216 (14)	77 (5-6)	107 (11-12)	116 (11)	58 (6)	110 (9)	84 (9)
	SVT	95 (6)	80 (8-9)	72 (9-10)	87 (4)	154 (9)	98 (6)	74 (11)	180 (9)	78 (7-9)	75 (4)	102 (8)	57 (5)	108 (8)	82 (7)
	TeNMF	104 (9)	93 (12)	386 (15)	131 (11)	166 (13)	176 (15)	52 (7)	214 (13)	77 (5-6)	105 (10)	112 (10)	43 (2)	113 (10-11)	78 (6)
Deep learning	BRITS	116 (14)	96 (13)	35 (2)	182 (14)	120 (6)	110 (9)	86 (14)	201 (12)	168 (15)	74 (3)	132 (14)	67 (11-13)	139 (13)	103 (13)
	GP-VAE	91 (5)	68 (5)	61 (7)	225 (15)	78 (4)	105 (8)	41 (5-6)	155 (6-7)	114 (13)	86 (6)	99 (7)	67 (11-13)	149 (14)	108 (14)
	M-RNN	167 (16)	105 (15)	129 (13)	168 (13)	165 (12)	147 (14)	77 (13)	155 (6-7)	132 (14)	134 (15)	124 (13)	137 (15)	164 (15)	150 (15)
	NAOMI	76 (2-3)	49 (4)	49 (5)	114 (6-7)	80 (5)	74 (4)	41 (5-6)	128 (4)	86 (12)	107 (11-12)	90 (5)	65 (10)	**9 (1)**	37 (3)
	SAITS	79 (4)	45 (2-3)	45 (3-4)	**54 (1-2)**	**61 (1-2)**	57 (3)	**24 (1-2)**	101 (1-3)	72 (2-3)	**63 (1-2)**	**65 (1-2)**	53 (3-4)	17 (2)	**35 (1)**
	Transformer	76 (2-3)	45 (2-3)	45 (3-4)	**54 (1-2)**	**61 (1-2)**	56 (2)	**24 (1-2)**	101 (1-3)	72 (2-3)	**63 (1-2)**	**65 (1-2)**	53 (3-4)	18 (3)	36 (2)
	SANNI	**56 (1)**	**29 (1)**	**33 (1)**	71 (3)	76 (3)	**53 (1)**	25 (3)	101 (1-3)	**63 (1)**	125 (14)	78 (3)	**39 (1)**	76 (4)	58 (4)
Overlap															
Unsupervised	CDRec	104 (8)	99 (9)	96 (9)	150 (6-7)	163 (12-13)	122 (6)	56 (5-6)	248 (12)	50 (7)	94 (10)	112 (8)	15 (2)	147 (8)	81 (4)
	DynaMMO	117 (11-12)	82 (2)	88 (8)	140 (4)	140 (8)	113 (4)	48 (2-3)	313 (16)	47 (3)	83 (5)	123 (10)	31 (7)	133 (7)	82 (5)
	GROUSE	112 (10)	317 (14)	254 (13)	447 (16)	284 (16)	283 (16)	225 (15)	243 (9)	268 (15)	147 (14)	221 (14)	137 (15)	173 (10)	155 (13)
	ORBITS	96 (5)	89 (4)	401 (14)	162 (9)	168 (14)	183 (13)	56 (5-6)	209 (5)	76 (11)	97 (11)	110 (6)	77 (12)	125 (3)	101 (9)
	ROSL	110 (9)	96 (6-7)	65 (4)	150 (6-7)	102 (5)	105 (2)	58 (7)	220 (6)	72 (10)	67 (4)	104 (5)	39 (9)	129 (5-6)	84 (6)
	SoftImp.	100 (6)	83 (3)	84 (6)	151 (8)	150 (9)	114 (5)	**46 (1)**	222 (7)	55 (8)	93 (9)	104 (4)	28 (6)	**117 (1)**	72 (2)
	SVDImp.	103 (7)	96 (6-7)	170 (12)	149 (5)	157 (10)	135 (10)	48 (2-3)	252 (13)	49 (5-6)	91 (8)	110 (7)	47 (10)	129 (5-6)	88 (7)
	SVT	94 (4)	91 (5)	86 (7)	164 (11)	90 (4)	105 (3)	52 (4)	227 (8)	48 (4)	62 (3)	97 (3)	**11 (1)**	118 (2)	**64 (1)**
	TeNMF	117 (11-12)	174 (12)	591 (15)	166 (12)	158 (11)	241 (14)	658 (16)	245 (11)	49 (5-6)	85 (6-7)	259 (16)	36 (8)	214 (11)	125 (10)
Deep learning	BRITS	139 (13)	141 (11)	102 (10)	163 (10)	103 (6)	130 (7)	96 (13)	291 (15)	180 (14)	105 (12)	168 (13)	70 (11)	223 (12)	146 (12)
	GP-VAE	283 (16)	97 (8)	75 (5)	267 (15)	176 (15)	180 (12)	93 (12)	186 (2)	90 (12)	85 (6-7)	114 (9)	90 (13)	249 (14)	170 (14)
	M-RNN	199 (15)	136 (10)	105 (11)	225 (13)	108 (7)	155 (11)	70 (11)	**177 (1)**	156 (13)	203 (15)	152 (12)	131 (14)	259 (15)	195 (15)
	NAOMI	143 (14)	198 (13)	603 (16)	236 (14)	163 (12-13)	269 (15)	139 (14)	244 (10)	311 (16)	204 (16)	224 (15)	180 (16)	676 (16)	428 (16)
	SAITS	86 (3)	365 (15-16)	61 (2-3)	112 (2-3)	**37 (1-2)**	132 (9)	64 (8-9)	200 (3-4)	**43 (1-2)**	**56 (1-2)**	**91 (1-2)**	26 (4-5)	128 (4)	77 (3)
	Transformer	83 (2)	365 (15-16)	61 (2-3)	112 (2-3)	**37 (1-2)**	132 (8)	64 (8-9)	200 (3-4)	**43 (1-2)**	**56 (1-2)**	**91 (1-2)**	26 (4-5)	167 (9)	96 (8)
	SANNI	**81 (1)**	**68 (1)**	**59 (1)**	**67 (1)**	62 (3)	**67 (1)**	65 (10)	258 (14)	62 (9)	122 (13)	127 (11)	23 (3)	244 (13)	134 (11)

Method		Dataset													
		Category A: Actor with activities						*Category B: Seasonality and/or cyclicity*					*Category C: Stochasticity*		
Type	Name	Electricity	Madrid	PAMAP	NREL	WalkRun	*Mean*	BAFU	MAREL	Meteo	Saaleaue	*Mean*	BTC	Soccer	*Mean*
							Disjoint								
Unsupervised	CDRec	93 (4)	89 (9)	159 (12)	146 (9-10)	127 (13)	123 (9)	53 (6)	538 (16)	52 (5-7)	96 (11)	185 (14)	12 (2)	140 (9)	76 (7)
	DynaMMO	116 (12)	84 (5-6)	97 (9)	127 (4)	113 (9)	107 (7)	46 (4)	227 (7)	50 (3-4)	71 (6)	98 (5)	13 (3)	129 (7)	71 (4)
	GROUSE	112 (11)	353 (16)	171 (14)	442 (16)	270 (16)	270 (15)	210 (16)	230 (9)	271 (16)	149 (15)	215 (16)	127 (16)	155 (11)	141 (12-13)
	ORBITS	96 (6)	90 (10)	287 (15)	148 (11)	129 (14)	150 (13)	56 (7-9)	211 (6)	71 (11)	101 (13)	110 (8)	45 (12)	122 (5)	84 (9)
	ROSL	111 (10)	91 (11)	72 (5)	141 (6-7)	85 (5-6)	100 (5)	91 (12)	238 (10)	60 (10)	59 (4)	112 (10)	14 (4)	112 (2)	**63 (1-2)**
	SoftImp.	99 (7)	85 (7)	77 (7)	141 (6-7)	118 (10)	104 (6)	45 (2-3)	229 (8)	57 (9)	92 (10)	106 (6)	16 (5)	117 (3-4)	66 (3)
	SVDImp.	102 (8)	86 (8)	161 (13)	146 (9-10)	122 (11)	123 (10)	47 (5)	255 (11)	52 (5-7)	91 (9)	111 (9)	18 (7)	128 (6)	73 (6)
	SVT	95 (5)	84 (5-6)	74 (6)	145 (8)	47 (3)	89 (4)	**43 (1)**	206 (3)	55 (8)	55 (3)	**90 (1)**	**9 (1)**	117 (3-4)	**63 (1-2)**
	TeNMF	104 (9)	96 (12)	115 (10)	163 (12)	125 (12)	121 (8)	45 (2-3)	259 (12)	52 (5-7)	76 (7)	108 (7)	30 (9)	134 (8)	82 (8)
Deep learning	BRITS	121 (13)	125 (13)	144 (11)	136 (5)	106 (8)	126 (11)	96 (13)	361 (15)	170 (14)	81 (8)	177 (13)	27 (8)	364 (16)	196 (16)
	GP-VAE	228 (16)	78 (4)	69 (4)	252 (15)	171 (15)	160 (14)	179 (15)	305 (14)	94 (13)	98 (12)	169 (12)	67 (14)	269 (15)	168 (15)
	M-RNN	127 (14)	144 (14)	92 (8)	197 (13)	85 (5-6)	129 (12)	63 (11)	**172 (1)**	87 (12)	60 (5)	96 (4)	99 (15)	183 (13)	141 (12-13)
	NAOMI	178 (15)	255 (15)	310 (16)	238 (14)	86 (7)	213 (16)	152 (14)	201 (2)	206 (15)	182 (16)	185 (15)	56 (13)	241 (14)	148 (14)
	SAITS	90 (2)	60 (2-3)	55 (2-3)	115 (2-3)	**30 (1-2)**	70 (2-3)	56 (7-9)	210 (4-5)	**47 (1-2)**	**51 (1-2)**	91 (2-3)	35 (10-11)	**109 (1)**	72 (5)
	Transformer	92 (3)	60 (2-3)	55 (2-3)	115 (2-3)	**30 (1-2)**	70 (2-3)	56 (7-9)	210 (4-5)	**47 (1-2)**	**51 (1-2)**	91 (2-3)	35 (10-11)	144 (10)	90 (11)
	SANNI	**68 (1)**	**50 (1)**	**49 (1)**	**65 (1)**	51 (4)	**57 (1)**	58 (10)	281 (13)	50 (3-4)	142 (14)	133 (11)	17 (6)	160 (12)	88 (10)

ACKNOWLEDGMENTS

The research is carried out using the supercomputer resources of South Ural State University (Chelyabinsk, Russia).

FUNDING

This work was financially supported by the Russian Science Foundation (grant no. 23-21-00465).

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. S. Kumar, P. Tiwari, and M. L. Zymbler, "Internet of things is a revolutionary approach for future technology enhancement: A review," J. Big Data **6**, 111 (2019). https://doi.org/10.1186/s40537-019-0268-2
2. S. Ivanov, K. Nikolskaya, G. Radchenko, L. Sokolinsky, and M. Zymbler, "Digital twin of city: Concept overview," in *Proceedings of the 2020 Global Smart Industry Conference GloSIC 2020, Chelyabinsk, Russia, November 17–19, 2020,* pp. 178–186. https://doi.org/10.1109/GloSIC50886.2020.9267879
3. M. Zymbler, Y. Kraeva, E. Latypova, S. Kumar, D. Shnayder, and A. Basalaev, "Cleaning sensor data in smart heating control system," in *Proceedings of the 2020 Global Smart Industry Conference GloSIC 2020, Chelyabinsk, Russia, November 17–19, 2020,* pp. 375–381. https://doi.org/10.1109/GloSIC50886.2020.9267813
4. M. Khayati, I. Arous, Z. Tymchenko, and P. Cudré-Mauroux, "ORBITS: Online recovery of missing values in multiple time series streams," Proc. VLDB Endow. **14**, 294–3060 (2020). http://www.vldb.org/pvldb/vol14/p294-khayati.pdf
5. M. Khayati, A. Lerner, Z. Tymchenko, and P. Cudré-Mauroux, "Mind the gap: An experimental evaluation of imputation of missing values techniques in time series," Proc. VLDB Endow. **13**, 768–782 (2020). http://www.vldb.org/pvldb/vol13/p768-khayati.pdf
6. C. Fang and C. Wang, "Time series data imputation: A survey on deep learning approaches," arXiv: 2011.11347 (2020). https://arxiv.org/abs/2011.11347

7. J. Wang, W. Du, W. Cao, K. Zhang, W. Wang, Y. Liang, and Q. Wen, "Deep learning for multivariate time series imputation: A survey," arXiv: 2402.04059 (2024). https://doi.org/10.48550/arXiv.2402.04059
8. B. D. Minor, J. R. Doppa, and D. J. Cook, "Learning activity predictors from sensor data: Algorithms, evaluation, and applications," IEEE Trans. Knowl. Data Eng. **29**, 2744–2757 (2017). https://doi.org/10.1109/TKDE.2017.2750669
9. S. Imani, F. Madrid, W. Ding, S. E. Crouter, and E. J. Keogh, "Introducing time series snippets: A new primitive for summarizing long time series," Data Min. Knowl. Discov. **34**, 1713–1743 (2020). https://doi.org/10.1007/s10618-020-00702-y
10. A. Yurtin and M. Zymbler, SANNI: Snippet and artificial neural network-based imputation of missing values in multivariate time series. https://github.com/yurtinaa/SANNI. Accessed January 8, 2024.
11. M. Khayati, P. Cudré-Mauroux, and M. H. Böhlen, "Scalable recovery of missing blocks in time series with high and low cross-correlations," Knowledge Inform. Syst. **62**, 2257–2280 (2020). https://doi.org/10.1007/s10115-019-01421-7
12. L. Li, J. McCann, N. S. Pollard, and C. Faloutsos, "DynaMMo: Mining and summarization of coevolving sequences with missing values," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28–July 1, 2009,* Ed. by F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki (ACM, 2009), pp. 507–516. https://doi.org/10.1145/1557019.1557078
13. X. Shu, F. Porikli, and N. Ahuja, "Robust orthonormal subspace learning: Efficient recovery of corrupted low-rank matrices," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition CVPR 2014, Columbus, OH, June 23–28, 2014* (IEEE Comput. Soc., 2014), pp. 3874–3881. https://doi.org/10.1109/CVPR.2014.495
14. L. Balzano, Y. Chi, and Y. M. Lu, "Streaming PCA and subspace tracking: The missing data case," Proc. IEEE **106**, 1293–1310 (2018). https://doi.org/10.1109/JPROC.2018.2847041
15. R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," J. Mach. Learn. Res. **11**, 2287–2322 (2010). https://dl.acm.org/doi/10.5555/1756006.1859931
16. O. G. Troyanskaya, M. N. Cantor, G. Sherlock, P. O. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," Bioinform. **17**, 520–525 (2001). https://doi.org/10.1093/bioinformatics/17.6.520
17. J. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," SIAM J. Optim. **20**, 1956–1982 (2010). https://doi.org/10.1137/080738970
18. J. Mei, Y. de Castro, Y. Goude, and G. Hébrail, "Nonnegative matrix factorization for time series recovery from a few temporal aggregates," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017,* Proc. Mach. Learn. Res. **70**, 2382–2390 (2017). http://proceedings.mlr.press/v70/mei17a.html
19. Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E^2GAN: End-to-end generative adversarial network for multivariate time series imputation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence IJCAI 2019, Macao, China, August 10–16, 2019,* Ed. by S. Kraus (2019), pp. 3094–3100. https://doi.org/10.24963/ijcai.2019/429
20. Z. Wu, C. Ma, X. Shi, L. Wu, D. Zhang, Y. Tang, and M. Stojmenovic, "BRNN-GAN: Generative adversarial networks with bi-directional recurrent neural networks for multivariate time series imputation," in *Proceedings of the 27th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2021, Beijing, China, December 14–16, 2021* (IEEE, 2021), pp. 217–224. https://doi.org/10.1109/ICPADS53394.2021.00033
21. W. Du, D. Côté, and Y. Liu, "SAITS: Self-attention-based imputation for time series," Expert Syst. Appl. **219**, 119619 (2023). https://doi.org/10.1016/j.eswa.2023.119619
22. E. Oh, T. Kim, Y. Ji, and S. Khyalia, "STING: Self-attention based time-series imputation networks using GAN," in *Proceedings of the IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7–10, 2021,* Ed. by J. Bailey, P. Miettinen, Y. S. Koh, D. Tao, and X. Wu (IEEE, 2021), pp. 1264–1269. https://doi.org/10.1109/ICDM51629.2021.00155
23. Y. Liu, R. Yu, S. Zheng, E. Zhan, and Y. Yue, "NAOMI: Non-autoregressive multiresolution sequence imputation," in *Advances in Neural Information Processing Systems 32, Annual Conference 2019 NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada,* Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett (2019), pp. 11236–11246. https://proceedings.neurips.cc/paper/2019/hash/50c1f44e426560f3f2cdcb3e19e39903-Abstract.html
24. V. Fortuin, D. Baranchuk, G. Rätsch, and S. Mandt, "GP-VAE: Deep probabilistic time series imputation," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, August 26–28, 2020, Palermo, Sicily, Italy,* Ed. by S. Chiappa and R. Calandra, Proc. Mach. Learn. Res. **108**, 1651–1661 (2020). http://proceedings.mlr.press/v108/fortuin20a.html

25. 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 31: Annual Conference NeurIPS 2018, December 3–8, 2018, Montréal, Canada,* Ed. by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (2018), pp. 6776–6786. https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html
26. J. Yoon, W. R. Zame, and M. van der Schaar, "Estimating missing data in temporal data streams using multi-directional recurrent neural networks," IEEE Trans. Biomed. Eng. **66**, 1477–1490 (2019). https://doi.org/10.1109/TBME.2018.2874712
27. J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6–11, 2015,* Ed. by F. R. Bach and D. M. Blei, JMLR Workshop Conf. Proc. **37**, 2067–2075 (2015). http://proceedings.mlr.press/v37/chung15.html
28. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
29. S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," Int. J. Uncertain. Fuzziness Knowl. Based Syst. **6**, 107–116 (1998). https://doi.org/10.1142/S0218488598000094
30. A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks **18**, 602–610 (2005). https://doi.org/10.1016/j.neunet.2005.06.042
31. S. Gharghabi, S. Imani, A. J. Bagnall, A. Darvishzadeh, and E. J. Keogh, "An ultra-fast time series distance measure to allow data mining in more complex real-world deployments," Data Min. Knowl. Discov. **34**, 1104–1135 (2020). https://doi.org/10.1007/ s10618-020-00695-8
32. M. Zymbler and A. Goglachev, "Fast summarization of long time series with graphics processor," Mathematics **10**, 1781 (2022). https://doi.org/10.3390/math10101781
33. A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Proceedings of the 16th International Symposium on Wearable Computers, ISWC 2012, Newcastle, UK, June 18–22, 2012* (IEEE Comput. Soc., 2012), pp. 108–109. https://doi.org/10.1109/ISWC.2012.13
34. C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. J. Keogh, "Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," in *Proceedings of the IEEE 16th International Conference on Data Mining ICDM 2016, December 12–15, 2016, Barcelona, Spain* (ICDM, 2016), pp. 1317–1322. https://doi.org/10.1109/ICDM.2016.0179
35. L. Lu, Y. YeonjongSu, and G. Em Karniadakis, "Dying ReLU and initialization: Theory and numerical examples," Commun. Comput. Phys. **28**, 1671–1706 (2020). http://global-sci.org/intro/article_detail/cicp/18393.html
36. A. Trindade, "Electricity load diagrams 2011–2014," UCI Machine Learning Reposit. (2015). https://doi.org/10.24432/C58C86
37. I. Laña, I. Olabarrieta, M. Vélez, and J. Del Ser, "On the imputation of missing data for road traffic forecasting: New insights and novel techniques," Transp. Res., Part C: Emerg. Technol. **90**, 18–33 (2018). https://doi.org/10.1016/j.trc.2018.02.021
38. M. Sheppy, A. Beach, and S. Pless, NREL RSF Measured Data 2011. https://data.openei.org/submissions/358 (2014). Accessed March 09, 2023.
39. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. https://www.hydrodaten.admin.ch/. Accessed March 09, 2023.
40. A. C. Lozano, H. Li, A. Niculescu-Mizil, Y. Liu, C. Perlich, J. R. M. Hosking, and N. Abe, "Spatial-temporal causal modeling for climate change attribution," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28–July 1, 2009,* Ed. by J. F. Elder IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki (ACM, 2009), pp. 587–596. https://doi.org/10.1145/1557019.1557086
41. A. Lefebvre, MAREL Carnot data and metadata from Coriolis Data Centre, SEANOE, 2015. https://doi.org/10.17882/39754. Accessed March 09, 2023.
42. Meteoswiss: Federal Office of Meteorology and Climatology. https://www.meteoswiss.admin.ch/services-and-publications/service/open-government-data.html. Accessed March 09, 2023.
43. Weather Station Saaleaue, Max Planck Institute for Biogeochemistry, Germany. https://www.bgc-jena.mpg.de/wetter/weather_data.html. Accessed March 09, 2023.
44. Historical Crypto Data. https://www.cryptodatadownload.com/data/. Accessed March 09, 2023.

45. C. Mutschler, H. Ziekow, and Z. Jerzak, "The DEBS 2013 grand challenge," in *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems, DEBS'13, Arlington, TX, USA, June 29–July 03, 2013,* Ed. by S. Chakravarthy, S. D. Urban, P. R. Pietzuch, and E. A. Rundensteiner (ACM, 2013), pp. 289–294. https://doi.org/10.1145/2488222.2488283
46. N. Dolganina, E. Ivanova, R. Bilenko, and A. Rekachinsky, "HPC resources ofB South Ural State University," in *Proceedings of the 16th International Conference on Parallel Computational Technologies, PCT 2022, Dubna, Russia, March 29–31, 2022,* Ed. by L. Sokolinsky and M. Zymbler, Commun. Comput. Inform. Sci. **1618**, 43–55 (2022). https://doi.org/10.1007/978-3-03 1-11623-0_4
47. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference, December 4–9, 2017, Long Beach, CA, USA,* Ed. by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett (2017), pp. 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
48. Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, "High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals, and enabling technologies," IEEE Internet Things J. **6**, 7946–7970 (2019). https://doi.org/10.1109/JIOT.2019.2907245
49. H. Zhu, Z. Pang, B. Xie, and G. Bag, "IETF iot based wireless communication for latency-sensitive use cases in building automation," in *Proceedings of the 25th IEEE International Symposium on Industrial Electronics, ISIE 2016, Santa Clara, CA, June 8–10, 2016* (IEEE, 2016) pp. 1168–1173. https://doi.org/10.1109/ISIE.2016.7745060
50. K. Yu, Z. Pang, M. Gidlund, J. Åkerberg, and M. Björkman, "REALFLOW: Reliable real-time flooding-based routing protocol for industrial wireless sensor networks," Int. J. Distrib. Sens. Networks **10**, (2014). https://doi.org/10.1155/2014/936379
51. S. Alaee, R. Mercer, K. Kamgar, and E. J. Keogh, "Time series motifs discovery under DTW allows more robust discovery of conserved structure," Data Min. Knowledge Discov. **35**, 863–910 (2021). https://doi.org/10.1007/s10618-021-00740-0
52. A. Mueen, E. J. Keogh, Q. Zhu, S. Cash, and M. B. Westover, "Exact discovery of time series motifs," in *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30–May 2, 2009, Sparks, NV* (SIAM, 2009), pp. 473–484. https://doi.org/10.1137/1.9781611972795.41