SSIM—A Deep Learning Approach for Recovering Missing Time Series Sensor Data

Yi-Fan Zhang, *Member, IEEE*, Peter J. Thorburn, Wei Xiang, *Senior Member, IEEE*, and Peter Fitch

***Abstract*—Missing data are unavoidable in wireless sensor networks, due to issues such as network communication outage, sensor maintenance or failure, etc. Although a plethora of methods have been proposed for imputing sensor data, limitations still exist. First, most methods give poor estimates when a consecutive number of data are missing. Second, some methods reconstruct missing data based on other parameters monitored simultaneously. When all the data are missing, these methods are no longer effective. Third, the performance of deep learning methods relies highly on a massive number of training data. Moreover in many scenarios, it is difficult to obtain large volumes of data from wireless sensor networks. Hence, we propose a new sequence-to-sequence imputation model (SSIM) for recovering missing data in wireless sensor networks. The SSIM uses the state-of-the-art sequence-to-sequence deep learning architecture, and the long short-term memory network is chosen to utilize both past and future information for a given time. Moreover, a variable-length sliding window algorithm is developed to generate a large number of training samples so the SSIM can be trained with small data sets. We evaluate the SSIM by using real-world time series data from a water quality monitoring network. Compared to methods like ARIMA, seasonal ARIMA, matrix factorization, multivariate imputation by chained equations, and expectation–maximization, the proposed SSIM achieves up to 69.2%, 70.3%, 98.3%, and 76% improvements in terms of the root mean square error, mean absolute error, mean absolute percentage error (MAPE), and symmetric MAPE, respectively, when recovering missing data sequences of three different lengths. The SSIM is therefore a promising approach for data quality control in wireless sensor networks.**

***Index Terms*—Deep learning, imputation, sequence-to-sequence, time series, wireless sensor networks.**

I. Introduction

Wireless sensor networks are in widespread use in various application areas such as agriculture, industry, and environment. High frequency measurements can reveal complex temporal dynamics that may be obscured by traditional sampling methods, and offer new insights into the inner workings of a monitored system [1]. However, the issue of missing data is relatively common in wireless sensor networks and can have a negative effect on the conclusions drawn from the data [2], [3].

Manuscript received February 10, 2019; accepted March 19, 2019. Date of publication April 3, 2019; date of current version July 31, 2019. This work was supported by the CSIRO Digiscape Future Science Platform. *(Corresponding author: Yi-Fan Zhang.)*

Y.-F. Zhang and P. J. Thorburn are with the Department of Agriculture and Food, CSIRO, Brisbane, QLD 4067, Australia (e-mail: yi-fan.zhang@csiro.au; peter.thorburn@csiro.au).

W. Xiang is with the College of Science and Engineering, James Cook University, Cairns, QLD 4878, Australia (e-mail: wei.xiang@jcu.edu.au).

P. Fitch is with the Department of Land and Water, CSIRO, Canberra, ACT 2601, Australia (e-mail: peter.fitch@csiro.au).

Digital Object Identifier 10.1109/JIOT.2019.2909038

Data imputation is a process for constructing missing values based on known data points. Using statistics (e.g., mean, median, or mode) for imputation can obtain good estimates of missing data when a time series has few discrete missing values. However, it is difficult to recover the consecutive missing sensory data.

Considering that a wireless sensor network usually monitors multiple parameters, another commonly used approach is to reconstruct the missing parameter based on other available parameters in the data sets. For example, Zhou and Huang [4] proposed an iterative imputing network to recover missing sensor data. In their approach, all supportive parameters are available and the authors do not consider missing data blocks. An auto-encoder neural network is proposed by Leke *et al.* [5] to predict missing data in the forest fire data set from the UCI data repository [6]. In many practical systems, all monitored parameters tend to get lost in the same period of time due to network or power outage issues. In this event, one can no longer predict the missing data points based on other parameters.

On the other hand, deep learning methods have been widely used in recovering missing sensor data in recent years [7], [8]. Though deep neural network models exhibit outstanding performance for wide-ranging data sets, the data-driven modeling approach necessitates huge volumes of training data. However, due to high costs of sensor probes, poor working conditions, or limited power supply, most wireless sensor networks only collect time series data with limited time indexes. For these systems, traditional deep learning techniques are ineffective.

Sliding window algorithms are prevalent in generating time series training samples for deep learning models [9]–[14]. The generic sliding window (GSW) algorithm generates samples with historical information as inputs, while the sliding window algorithm with future information (SWF) approach also includes observations from future time steps into each sample. Most recurrent neural network (RNN)-based data imputation models [9]–[11] use GSW for generating training samples. In addition, SWF is employed in some data-driven models [12]–[14] so the information before and after a missing reading's time index can both be utilized. An additional attribute of these approaches is that the size of the sliding window is fixed. When a wireless sensor network has not accumulated sufficient observations, the performance of deep learning models is inevitably affected by the insufficient number of training samples yielded by the GSW and SWF algorithms. Thus, a sliding window algorithm with a variable-length window size is highly desirable in processing time series data sets with small numbers of measurements.

Recently, sequence-to-sequence models in deep learning have achieved state-of-the-art results in many areas such as machine translation [15], image caption generation [16], and speech recognition [17]. The general architecture of the sequence-to-sequence model allows one to generate arbitrary target data from input data of variable length, which is suitable for recovering missing data sequences. In addition, most implementations of the sequence-to-sequence models employ long short-term memory network (LSTM) [18] layers as hidden units. This architecture is able to utilize more contexts and previous sequence regulations, which is important in processing time series sensor data.

In this paper, we propose a sequence-to-sequence imputation model (SSIM) for recovering variable-length missing data sequences in wireless sensor networks. The main contributions of this paper include the following.

1) We design a deep imputation model with a new sequence-to-sequence architecture. To our best knowledge, no other sequence-to-sequence model has been built for sensor data imputation to date. Our model is able to recover missing data sequences based on the information from both past and future time indexes.
2) We develop a variable-length sliding window (VLSW) algorithm to enhance training sample generation. Compared to traditional sliding window algorithms, the VLSW algorithm is able to yield a larger number of training samples from the same data sets. This allows us to train deep learning models based on small time series data sets, which significantly extends the applicability and versatility of deep learning models.

II. Problem Overview and Formulation

In most wireless sensor networks, sensors work collaboratively to monitor various characteristics of a target system. For example, electrode-based measurements of pH, conductivity, temperature or other parameters are now common in stream and watershed monitoring systems [1]. These parameters can give us a good understanding of the physical, chemical, and biological characteristics of the hydrological system in question.

In these cases, the assumption is that the distributions of the missing and complete data are the same and the missing data can be predicted from the complete data [19]. This type of missing data mechanism is dubbed missing completely at random (MCAR). It is assumed by most of existing missing data imputation methods [20] and thus it is also assumed in this paper.

Fig. 1 illustrates a common scenario for missing time series sensor data. In this case, all the sensor data during the same period of time are missing. This can be caused by a variety of factors including unstable sensor power supply, data transmission errors or regular device maintenance.

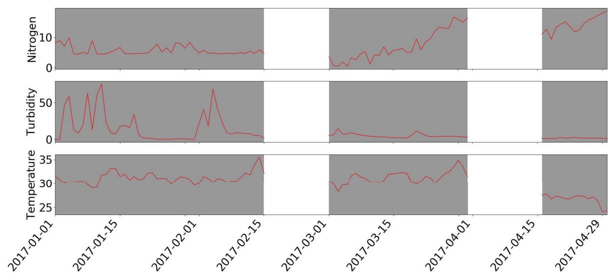


Fig. 1. Time series with continuous missing data. Gray blocks highlight the available time series data (red solid lines), while the white blank spaces represent the missing data sequences for different time series.

To formulate the problem, let us assume there are n numbers of time series generated by various sensor probes in a wireless sensor network under consideration. Each time series has numerical values with the same time interval. All the time series TS can be represented as follows:

$$\mathrm{TS}^T = \{\mathrm{ts}_1, \mathrm{ts}_2, \dots, \mathrm{ts}_n\}. \tag{1}$$

As shown in Fig. 1, partial consecutive data points in multiple periods of time are missing for all the time series TS. Let d_i and $\mathrm{ts}_i \backslash d_i$ denote the missing data and the remaining data in time series ts_i, respectively. Hence, all the available data in the wireless sensor network can be represented by

$$R^T = \{\mathrm{ts}_1 \backslash d_1, \mathrm{ts}_2 \backslash d_2, \dots, \mathrm{ts}_n \backslash d_n\} \tag{2}$$

where $\backslash$ represents the operation of removing elements from a sequence.

For a single time series ts_k, our goal is to obtain good estimates for the missing data sequence d_k by learning from all the available data R. In this problem, it is assumed the size of the remaining data is much bigger than that of the missing data, so that the chosen model will have enough information to learn

$$\forall i \in \{1, \dots, n\}, d_i = f(R) \tag{3}$$

where f is the function that the model is going to learn.

To solve the above problem, our idea is to first design a deep learning model so as to learn the temporal relationships among multiple time indexes and the n monitoring parameters. Second, we split the remaining data R into the training and validation sets and then generate a large number of training samples. Third, we train the proposed model and apply it to the test data for performance evaluation.

III. Proposed Sequence-to-Sequence Imputation Model

In this section, we propose a sequence-to-sequence model with the attention mechanism to solve the missing sensory data recovery problem formulated in Section II. The model is designed to accept and output variable-length data sequences, which is important in recovering a consecutive number of missing data in wireless sensor networks. Furthermore, a VLSW algorithm is designed to train our proposed model on small data sets, which significantly widens the scope and applicability of our proposed model.

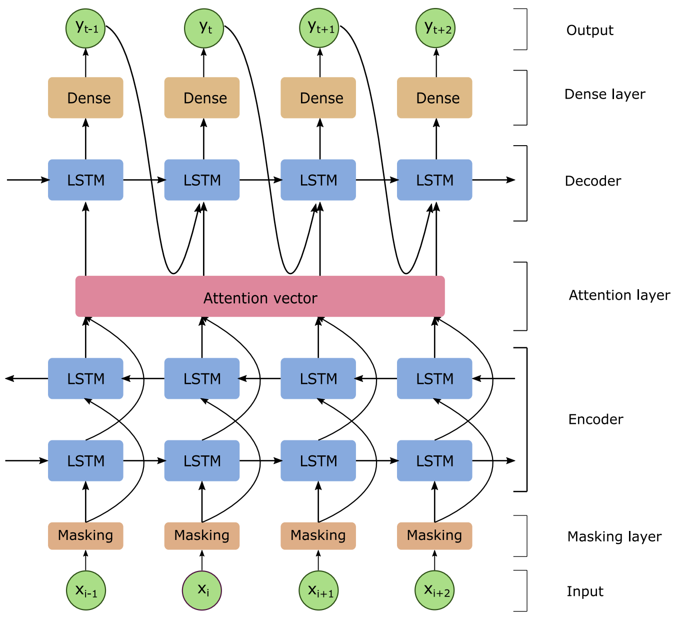


Fig. 2. SSIM architecture. The encoder in the SSIM is a BiLSTM, which is comprised of a forward LSTM and a backward LSTM. The decoder in the SSIM is an unidirectional LSTM. The dropout layers are also included in both encoder and decoder. A masking layer is added to remove the zero-padded vectors in the input sequences. A dense layer with linear activation function is stacked to product predictions with continues values.

A. SSIM

The SSIM utilizes the sequence-to-sequence architecture with the attention mechanism as depicted in Fig. 2, where the encoder and decoder are two key functional components. The encoder processes an input time series and maps it to a high-dimensional vector. The decoder takes input from the vector and yields target data sequences. Also, the attention mechanism enables the decoder learn how to focus on a specific range of the input sequence for the differing outputs.

1) BiLSTM Encoder: As opposed to the general prediction problem in which only past data are available, the missing data recovery problem formulated in Section II is different in the sense that one can use both past and future data points to reconstruct the missing data. Hence, we choose the bi-directional LSTM (BiLSTM) network as the encoder in our proposed SSIM.

The BiLSTM [21] extends the generic LSTM. It utilizes both previous and future context by processing the input data sequences from two directions with two separate LSTMs.

The general input of the LSTM is a variable-length sequence $x = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in \mathbb{R}^d$ and d represents the features in each time index i. In each time index, the LSTM maintains its internal hidden state h, which results in a hidden sequence of $\{h_1, h_2, \ldots, h_n\}$. The hidden vector h_t at time index t is updated as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{4}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{5}$$
$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{6}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{7}$$
$$h_t = o_t \otimes \tanh(c_t) \tag{8}$$

where i, f, and o refer to the input, forget, and output gates, respectively. c, σ, and $\otimes$ represent the cell vector, the sigmoid function, and the element-wise multiplication.

In a BiLSTM, instead of having only one sequence of h, the BiLSTM reads the input sequences from two directions to output two sequences with hidden states, namely: the forward states $\overrightarrow{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_n}\}$ and the backward states $\overleftarrow{h} = \{\overleftarrow{h_1}, \overleftarrow{h_2}, \ldots, \overleftarrow{h_n}\}$. Then, the hidden states at time index i are concatenated as follows:

$$h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]. \tag{9}$$

The output of the BiLSTM encoder is a sequence of hidden states $\{h_1, h_2, \ldots, h_n\}$, which has the same length n as the input sequence x.

2) LSTM Decoder With Attention: The decoder is responsible for recursively generating the output sequence $y = \{y_1, y_2, \ldots, y_m\}$. In this part, we choose the LSTM decoder with the attention mechanism similar to the global attention mechanism proposed by Luong *et al.* [22].

Unlike conventional sequence-to-sequence models designed for natural language processing, the data imputation problem under consideration in this paper requires outputs with continuous values. Hence, instead of calculating the probability distribution over next possible outputs, a fully connected layer with a linear activation function is added on top of the LSTM layer to produce predictions with continuous values. For the decoder, the estimated y_t at time t is computed as follows:

$$y_t = \text{Linear}(W[s_t; c_t] + b) \tag{10}$$
$$s_t = \text{LSTM}(y_{t-1}, s_{t-1}, c_t) \tag{11}$$

where s_t is the hidden state of the decoder at time index t, c_t is the attention context vector, and $[s_t; c_t]$ is a concatenation of the decoder hidden state and the context vector. The parameters W and b map the concatenation to the size of the decoder hidden states, and the linear layers product the final prediction y_t.

In each encoding time index t, the attention context vector c_t can be described as a weighted sum of the hidden states passed by the BiLSTM encoder

$$c_t = \sum_{i=1}^{n} \alpha_{ti} h_i. \tag{12}$$

The weight α_{ti} of each hidden states h_i is computed by

$$e_{ti} = a(s_{t-1}, h_i) \tag{13}$$
$$\alpha_{ti} = \text{softmax}(e_{ti}) \tag{14}$$

where e_{ti} represents the correlation between the hidden states around h_i and the output at time t. a is a feed-forward network that can be jointly trained with other components of the LSTM Decoder. A softmax activation function is applied to e_{ti} to ensure that the sum of all the attention weights is normalized to 1.

Intuitively, this implements a mechanism of attention in the decoder. The decoder decides parts of the input sequence it pays attention to. By giving the decoder an attention mechanism, the information can be spread throughout the data

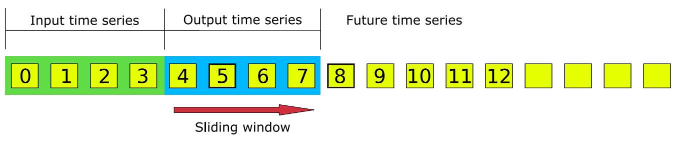


Fig. 3. Illustration of the GSW algorithm. The green and blue blocks represent the input and output time series, respectively. One training sample is comprised of the two time series.

sequence, which can be selectively retrieved by the decoder accordingly [23].

As can be observed from Fig. 2, after taking input from the time series data, the model's masking layers filter out the zero-padded vectors in the input data samples and pass them to the BiLSTM encoder. The encoder processes data in each time index along both forward and backward directions, and generates a hidden output vector. The hidden vector is then processed by the LSTM decoder and the target sequence is generated recursively by the final dense layer. During this process, the decoder calculates the attention weights in each time index so that it can focus on specific parts of the input to obtain relevant information.

B. Variable-Length Sliding Window Algorithm

When using deep learning models for processing time series data, an important task is to extract subsequences as training and testing samples from the original long time series. In this section, we will discuss two commonly adopted sliding window algorithms and then propose a new VLSW algorithm.

1) Generic Sliding Window Algorithm: In order to generate training samples, one first choose the lengths of the input and output sequences as desired. After that, a window with a fixed length is constructed (e.g., the window length is eight in Fig. 3). The traditional sliding window algorithm works by anchoring the left side in the first time index of a time series, and then keep moving until the right side of the window reaches the end of the time series. During the moving, the training sample in each moving step is generated.

2) Sliding Window Algorithm With Future Information: When recovering the missing data, if we treat the output time series as the target data sequence we tend to reconstruct, it is clear that the information in the upcoming time indexes has not been included in each sample (blue boxes depicted in Fig. 3). In this approach, deep learning models can only learn to reconstruct the missing data based on the previous information (green block in Fig. 3). This means half of the useful information is not fully utilized.

Instead of using only the previous information, an improved sliding window algorithm is illustrated in Fig. 4, in which both time sequences before and after the missing data are included as the input time series.

Although the improved sliding window algorithm in Fig. 4 can generate samples with information from both past and future time indexes, a challenge comes about, when dealing with time series data with small numbers of time indexes. Let $|IL|$, $|IR|$, $|O|$, and $|T|$ denote the length of the input time series on the left, the input time series on the right, the output time series, and the original time sequence, respectively. The

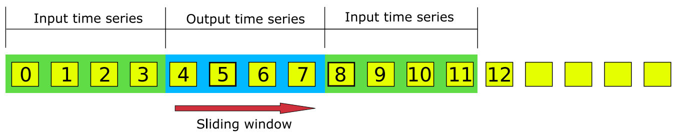


Fig. 4. Illustration of the SWF. The green and blue blocks represent the input and output time series, respectively. One training sample consists of the two time series.

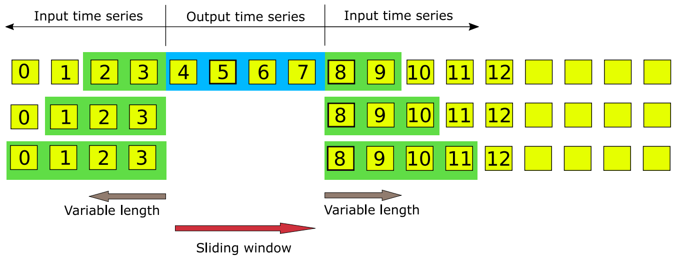


Fig. 5. Illustration of the proposed VLSW algorithm. The green and blue blocks represent the input and output time series, respectively. One training sample is composed of the two time series.

number of total samples N generated by the GSW and SWF algorithms can be expressed as follows:

$$N = |T| - |IL| - |O| - |IR| + 1$$
$$|IL| + |O| + |IR| \leq |T|. \tag{15}$$

As can be seen from (15), when $|T|$ is small, N also has a small value. When processing a time series with a small number of time indexes, these two algorithms cannot provide enough samples, which is essential for training any deep learning models.

3) Variable-Length Sliding Window Algorithm: In this paper, we propose a VLSW algorithm, which makes use of the information from both past and future time indexes. Furthermore, by dynamically changing the length of the input sequences in the process of sampling, the proposed VLSW algorithm can generate a much greater number of samples, which is important for implementing deep learning models.

Fig. 5 illustrates how the proposed VLSW algorithm works. In the VLSW algorithm, both $|IL|$ and $|IR|$ change during the sampling progress. Assume that $|IL|$ changes dynamically between m and n, while $|IR|$ changes dynamically between o and p. We have

$$\begin{aligned} N &= \sum_{i=m}^{n} \sum_{j=o}^{p} |T| - i - |O| - j + 1 \\ &= \frac{(p-o+1)(n-m+1)\big[2(|T|-|O|+1) - (m+n+o+p)\big]}{2} \\ &\quad n + p + |O| \leq |T| \end{aligned} \tag{16}$$

where m and n indicate the minimum and maximum lengths of $|IL|$, respectively, while o and p represent the minimum and maximum lengths of $|IR|$, respectively.

As can be observed from (15) and (16), the GSW and SWF algorithms are indeed two special cases of the proposed VLSW algorithm. We can derive the GSW algorithm by setting $o = p = 0$ and $m = n \geq 1$ in (16). Similarly, the SWF algorithm can be obtained by fixing $o = p \geq 1$ and $m = n \geq 1$ in (16).

Algorithm 1 VLSW Algorithm

Input: T for the original time series, m and o for the minimum lengths of the left and right input sequences, n and p for the maximum lengths of the left and right input sequences, and q for the length of the output sequence.

Output: S for all the samples. In S, S_{in} and S_{out} represent input and output sequences.

```
    Initialisation : $S \leftarrow \{(,)\}$, $IL, O, IR, M, s_{in}, s_{out} \leftarrow \{\}$,
    Sliding window:
 1: for $i = m$ to $n$ do
 2:     for $j = o$ to $p$ do
 3:         for $k = 1$ to $|T| - i - q - j + 1$ do
 4:             $IL \leftarrow \{T_k, \ldots, T_{k+i-1}\}$
 5:             $O \leftarrow \{T_{k+i}, \ldots, T_{k+i+q-1}\}$
 6:             $IR \leftarrow \{T_{k+i+q}, \ldots, T_{k+i+q+n-1}\}$
 7:             $M \leftarrow \{x|x = 0\}, |M| = |O|$
 8:             $s_{in} \leftarrow IL \cup M \cup IR$
 9:             $s_{out} \leftarrow O$
10:             $S \leftarrow \{(s_{in}, s_{out})\} \cup S$
11:         end for
12:     end for
13: end for
14: return $S$
```

Furthermore, (16) leads up to the ratio of the number of samples generated by the proposed VLSM algorithms to those generated by its GSW and SWF counterparts. We can formulate the ratio to

$$\frac{N_{\text{VLSM}}}{N_{\text{GSW}}} \approx \frac{N_{\text{VLSM}}}{N_{\text{SWF}}} \approx (p - o + 1)(n - m + 1) \tag{17}$$

where N_{VLSM}, N_{GSW}, and N_{SWF} represent the number of samples generated by the corresponding algorithms, respectively.

It should be taken note in Algorithm 1 that the VLSW algorithm pads a zero vector M between the *IL* and *IR* (lines 7 and 8) when generating the input sequence for each sample.

For example, a sample has input sequence $\{x_1, x_2, x_5, x_6\}$ and output sequence $\{x_3, x_4\}$. By padding zero vectors, one forces the SSIM to learn function f in a way described in the following equation:

$$\{\mathbf{x_3}, \mathbf{x_4}\} = f(x_1, x_2, \overrightarrow{0} \cdot \mathbf{x_3}, \overrightarrow{0} \cdot \mathbf{x_4}, x_5, x_6). \tag{18}$$

In this way, one can keep the original time structure for each sample. More specifically, the deep learning model would learn that *IL* ($\{x_1, x_2\}$) and *IR* ($\{x_5, x_6\}$) are not adjacent. In the SSIM architecture depicted in Fig. 2, a masking layer is added to ignore the zero vectors when calculating the hidden states.

4) Time Complexity Analysis: The time complexity of an algorithm is the total amount of time required by an algorithm to accomplish its tasks. A time-efficient algorithm is essential when used to solve large-scale practical problems. In this section, the time complexities of all the three sliding window algorithms are discussed.

As can be observed from Algorithm 1, the operations from lines 4 to 10 have the time complexity of $\mathcal{O}(1)$, the operation in line 14 has a time complexity of $\mathcal{O}(1)$, and the input/output variable initialization also has a time complexity of $\mathcal{O}(1)$. In addition, the loop in lines 1–3 would execute $n-m+1$, $p-o+1$, and $|T|-i-q-j+1$ times, respectively. Hence, considering the overall time complexity of the VLSW algorithm is determined by both size of the sliding window problem and execution time of the loop operations, we have

$$\begin{aligned}\mathcal{O} = \mathcal{O}((n - m + 1) * (p - o + 1) * (|T| - i - q - j + 1))\\ + 2 * \mathcal{O}(1).\end{aligned} \tag{19}$$

Considering (16), (19) can be rewritten as

$$\begin{aligned}\mathcal{O} = \mathcal{O}\left(\frac{(p - o + 1)(n - m + 1)\left[2(|T| - q + 1) - (m + n + o + p)\right]}{2}\right)\\ + 2 * \mathcal{O}(1).\end{aligned} \tag{20}$$

In the VLSW algorithm, the inputs n, m, o, p, and q are constants and their values are assigned before the execution of the algorithm. The size of the sliding window problem is determined by the length of the original time series $|T|$. Hence, the time complex $\mathcal{O}$ in (20) simplifies to

$$\mathcal{O} = \mathcal{O}(|T|) + 2 * \mathcal{O}(1) = \mathcal{O}(|T|). \tag{21}$$

As discussed in the previous section, the GSW and SWF algorithms are two special cases of the proposed VLSW algorithm. Therefore, we can derive the time complexities of the GSW and SWF algorithms in a similar manner, which can be shown to be $\mathcal{O}(|T|)$.

It is concluded that all the three algorithms have the same linear time complexity $\mathcal{O}(|T|)$. This means that the running time increases at most linearly with the length of the input for all the three sliding window algorithms, which is critical for processing large-scale time series data.

IV. Evaluation

In this section, we evaluate the effectiveness of the SSIM by using the water quality time series data collected by a government water quality monitoring program in Australia.

A. Time Series Data and the Monitoring Sensor Network

The Great Barrier Reef Catchment Loads Monitoring Program tracks long-term trends in water quality entering the Great Barrier Reef lagoon from adjacent catchments [24]. This involves monitoring 43 sites in 20 key catchments for sediment and nutrients and a further 19 sites for pesticides. Water quality data are integrated automatically into a cloud-based data monitoring platform via 4G networks. Each monitoring site deploys advanced water quality sensors. For example, the YSI EXO2 multiparameter sonde is installed for measuring the concentration of nitrogen. Also, weather stations deployed by Australian Bureau of Meteorology [25] were also used to provide rainfall information in this paper.

Data were collected from two monitoring sites in the Mulgrave-Russell catchment in the Great Barrier Reef, Australia (Table I). Water quality data usually exhibit characteristics such as non-normal distribution, presence of outliers or serial dependence, which can impact the performance of data-driven models significantly [26]. For example, the

TABLE I
WATER QUALITY DATA DURING JANUARY 1, 2017 AND MAY 1, 2018

Parameters (Daily)	Unit	Min	Max	Mean	Std Dev
Temperature max	°C	22.5	37	29.5	2.5
Temperature min	°C	12	27.5	21.2	2.8
Rainfall	mm	0	243	13.2	32.1
Evaporation	mm	2.2	29.8	5.9	2.1
Radiation	$\mathrm{MJ/m^2}$	6.0	30	18.1	5.1
Vapour pressure	hPa	12	32	24.9	4.0
Electrical conductivity	μS / cm	0.07	18545.3	1683.9	3106.7
Water discharge	$\mathrm{m^3/s}$	-11.3	851.9	47.4	97.9
Level	m	10	14	10.4	0.4
Turbidity	NTU	0.1	128.5	8.7	16.1
Nitrate	μM	0.1	37.2	13.8	6.4

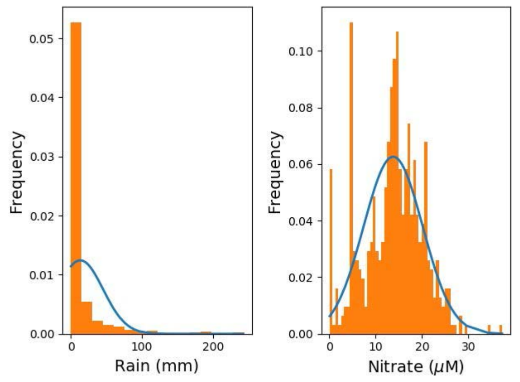


Fig. 6. Distribution of rainfall and nitrate concentrations during January 1, 2017 and May 1, 2018.

distribution of rainfall data during this observation period was highly skewed to the right (Fig. 6), which is affected by the wet/dry season pattern. Similarly, the distribution of nitrate is right skewed and a large number of observations are near zero. These attributes of the data pose great challenges to train data-driven deep learning models.

Among the above water quality parameters, nitrogen is known to be the most influential one affecting the health of the Great Barrier Reef, and the cost of measuring nitrogen is prohibitively high [27]. Hence, we choose to recover the missing nitrogen sensory data in the monitoring data sets. Our model is applied to the water quality data collected from January 1, 2017 to July 31, 2017 and September 1, 2017 to March 31, 2018 as the training set. The water quality data collected in August 2017 and April 2018 were chosen as the test set. Overall, the model uses 14-month training data and 2-month testing data.

B. Evaluation Metrics and Baseline Methods

We evaluate the performance of recovering missing data based upon the root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and symmetric MAPE (SMAPE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(|f_i - \hat{f}_i|)^2} \tag{22}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f_i - \hat{f}_i| \tag{23}$$

$$\mathrm{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{f_i - \hat{f}_i}{f_i}\right|\% \tag{24}$$

$$\mathrm{SMAPE} = \frac{200}{n}\sum_{i=1}^{n}\frac{f_i - \hat{f}_i}{|f_i| + |\hat{f}_i|}\% \tag{25}$$

where f_i and $\hat{f}_i$ are the true and estimated values of a parameter under monitoring, respectively.

Metrics suited for evaluating the accuracy of prediction vary in practice, and each metric has its own pros and cons [28]. In this paper, we employ two types of metrics. That is, the RMSE and MAE are scale-dependent metrics, while the MAPE and SMAPE are percentage-error metrics. The metrics are chosen based on the characteristics of the data imputation problem in question.

Despite issues such as high sensitivity to outliers, the RMSE is commonly used in statistical model evaluation [29]. The MAE reduces the impact of outlying observations. However, it cannot indicate the bias of the prediction, i.e., under-predicting or over-predicting [30]. For percentage-error metrics, the MAPE places a heavier penalty on forecasts that exceed the actual than those that are less than the actual. The SMAPE is defined as the symmetric MAPE to address the above issue. Nevertheless, it penalizes low forecasts more than high ones, and thus favors higher predictions than lower ones [31].

The water quality sensor data used in this experiment have its unique characteristics. First, several water quality parameters stay at low levels for a long period of time. Second, high concentration of nitrogen only occurs in short periods of time, but nitrogen is the most critical variable in water quality monitoring. Furthermore, outliers are inevitable in water quality monitoring thanks to environmental interference.

Hence, it is desired that the proposed data imputation model delivers a constant performance irrespective of the level of the nitrogen concentration. In addition, the model should avoid inherent prediction bias so it would not tend to produce negative or positive errors. Given this consideration, the RMSE and MAE can provide an assessment of the model's overall performance. In addition, the RMSE can also reveal whether the model's performance is affected by outliers. The value of the MAPE and SMAPE indicate whether the model has more negative ($y < \hat{y}$) or positive errors ($y > \hat{y}$). Overall, the use of all the aforementioned four metrics helps reveal a broader range of characteristics of the water quality data and provides a better insight into the SSIM's imputation accuracy.

In the experiments, we also compared our model with the following five data imputation methods. Among these methods, ARIMA [32] and SARIMA [33] are specifically designed for time series processing. The matrix factorization (MF) method in [34] imputes missing data based on matrix calculations. The multiple imputation by chained equations (MICE) method in [35] combines the simple imputation with regression models. The expectation–maximization (EM) scheme in [36] is a probabilistic imputation method.

C. Model Parameters

In order to use the proposed SSIM for recovering missing time series data, two important sets of parameters need to be determined: 1) the parameters for generating the training/testing samples and 2) the hyperparameters of the SSIM.

TABLE II
INITIALIZATION PARAMETERS FOR THE VLSW ALGORITHM

Parameter	Experimental cases		
	Case 1	Case 2	Case 3
m	2	2	2
n	7	7	7
o	2	2	2
p	7	7	7
q	5	6	7

TABLE III
HYPERPARAMETERS OF THE SSIM

Hyperparameters	Value
No. of Hidden Layers for Encoder	2
No. of Hidden Layers for Decoder	2
No. of Hidden LSTM Units per Layer	25
Dropout Rate	0.2
Optimizer	Adam
Loss Function	MSE

TABLE IV
NUMBER OF GENERATED TRAINING SAMPLES

Algorithm	No. of samples		
	Case 1	Case 2	Case 3
GSW	402	400	398
SWF	388	386	384
VLSW	14328	14256	14184

In the proposed VLSW algorithm listed in Algorithm 1, five parameters m, n, o, p, and q need to be initialized, representing the minimum and maximum lengths of the left input sequence, the minimum and maximum lengths of the right input sequence, and the length of the output sequence, respectively.

For the water quality sensory data collected from the Great Barrier Reef lagoon, the maximum period of missing values is only five days thanks to regular maintenance of the sensors. Hence, the horizon of prediction (q) is set to five in experimental case 1 to fit the data imputation problem under consideration in this paper. Furthermore, to evaluate the performance of the SSIM in recovering missing data of variable lengths, two extra experimental cases are added with output data sequences of lengths six and seven. All the parameters are listed in Table II.

The parameters in Table II are settled so as to obtain the weekly variation of water quality, which can bring valuable temporal information to the imputation methods. By using the parameter values in Table II, our proposed VLSW algorithm can generate a large number of samples for training and testing the SSIM. In addition, 10% of the data in the training is set to the validation set in each experimental case.

The structure of the SSIM also needs to be determined. The hyperparameters include the number of hidden layers, the number of hidden LSTM units in each layer, the dropout rate, the loss function, and so on. The SSIM is implemented and optimized by using the Keras neural network framework [37]. The optimized hyperparameters are shown in Table III.

The MSE is a mathematically well-behaved function, which is smoothly differentiable and easy to compute the gradient. Hence, the MSE is chosen as the loss function for training the SSIM.

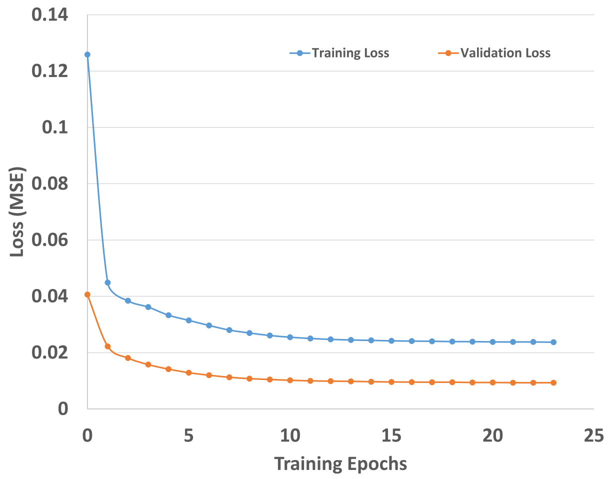


Fig. 7. Training and validation loss of the SSIM over 25 epochs. In this plot, the training loss is slightly higher than the validation loss because the dropout layer is actived during training.

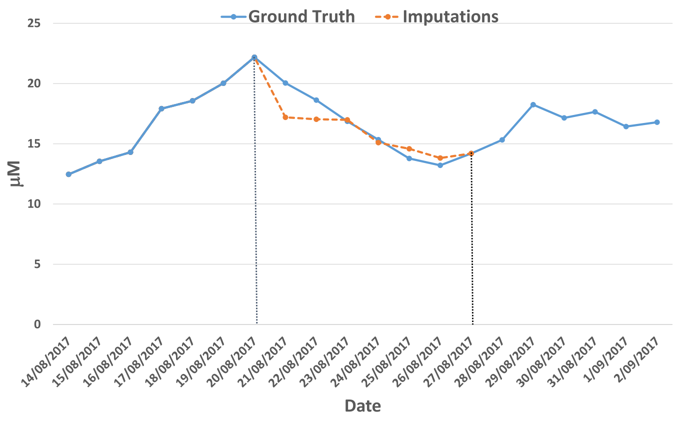


Fig. 8. Recovering six missing nitrogen data from August 21, 2017 to August 26, 2017 by using SSIM. Blue solid line and orange dotted line represent the ground truth data and the imputations generated by SSIM, respectively. The seven days' data comes before and after this period of time are used as inputs.

In our experiments, the model structure is identical in all the three experimental cases, which includes all the hyperparameters listed in Table III. Applying the same SSIM for all the three experimental cases ensures a fair comparison among the comparative data imputation methods.

However, in practical applications, both VLSW's configurations and SSIM's hyperparameters vary in accordance with diverse data imputation problems. The settings of the VLSW algorithm should take into account the properties of the data sets in question. Furthermore, fine-tuning hyperparameters in consideration of each specific case can significantly improve the model's performance.

D. Experimental Results and Discussion

In the experiments, we first generate all the training samples using three sliding window algorithms discussed in Section III-B. In the comparison, the GSW algorithm generates samples with seven historical data points, the SWF algorithm generates samples with seven historical and seven future data points, while the proposed VLSW has initialized parameters

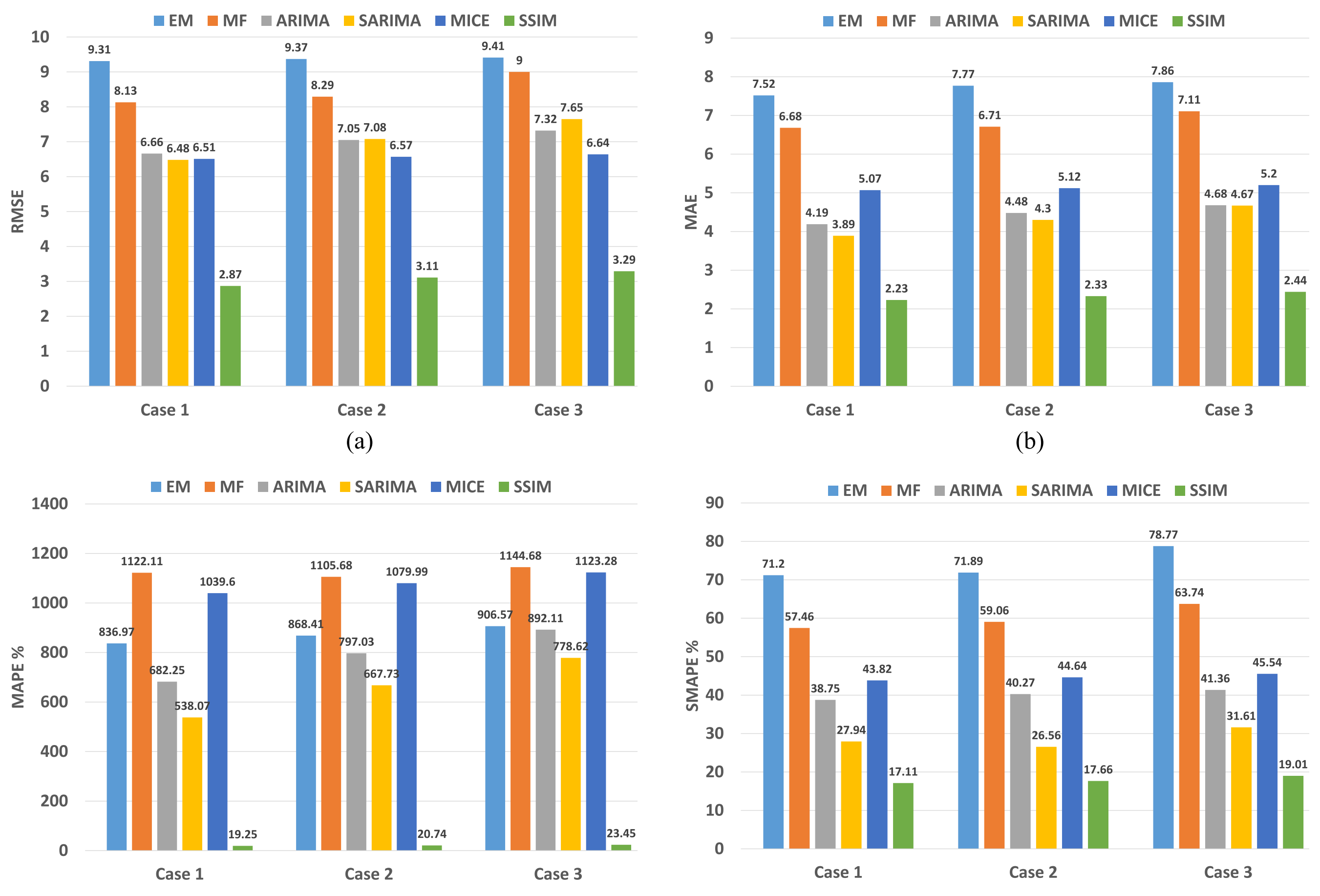


Fig. 9. Evaluation of missing data imputation using four metrics in three cases. (a) RMSE of six different methods in three cases. (b) MAE of six different methods in three cases. (c) MAPE of six different methods in three cases. (d) SMAPE of six different methods in three cases.

defined in Table II. The numbers of generated training samples for the three comparative algorithms are listed in Table IV.

As shown in Table IV, both GSW and SWF algorithms can only generate around 400 training samples, which is too small for training neural network models, especially when the models have deep structures.

By comparison, our proposed VLSW algorithm is able to generate over 14 000 training samples from the same time series data set, which is about 36 times higher than the comparative sliding window algorithms. This meets the ratio we derive from (17), and the number of training samples can be even greater if one adjusts the initialization parameters in Algorithm 1. This is because the VLSW algorithm can dynamically adjust the size of the sliding window in the process of samples generation. This ensures a significant number of samples for training the SSIM.

We trained the SSIM on a CSIRO's HPC server with 1 Core Xeon E5-2690 and 16-GB memory. Fig. 7 illustrates the variation of the training and validation losses through the first 25 epochs. It is apparent that the model's training and validation loss are both reduced dramatically in the first ten epochs, and then keep decreasing during the training process until the model converges. In this experiment, the best SSIM was obtained after around 1500 epochs. Each epoch was completed within 60 s.

A data imputation example is illustrated in Fig. 8. For the SSIM, the missing data points are predicted one by one from August 21, 2017 to August 26, 2017. Each time the model yields one output, it will combine this output with the previous inputs to generate the next new output, as illustrated in the decoder structure in Fig. 2. Generally, this predictive mechanism may accumulate predictive errors among time, resulting in bad predictions for the last few missing data.

The above issue is properly dealt with our proposed VLSW algorithm in conjunction with the BiLSTM encoder. In the VLSW algorithm, future data points are fed to the input time sequences. Hence, the hidden vector received by the decoder contains information from both past and future time indexes. Moreover, the BiLSTM encoder is able to exploit both forward and backward temporal dependencies. Hence, the prediction of the last few missing data points can be enhanced by processing the input time series in a reversed way. For example, though the trend of nitrogen has been decreasing since August 21, 2017, the imputed data on both August 25, 2017 and August 25, 2017 were not greatly reduced. This is because the imputation was also guided by the information from August 27, 2017 to September 2, 2017.

We also compared the SSIM with five different data imputation methods listed in Section IV-B. Four distinct performance metrics were used in measuring the imputation accuracy with

TABLE V
PERFORMANCE IMPROVEMENTS ACHIEVED BY THE SSIM

	Performance Improvements											
	Case 1				Case 2				Case 3			
	RMSE	MAE	MAPE	SMAPE	RMSE	MAE	MAPE	SMAPE	RMSE	MAE	MAPE	SMAPE
EM	69.2%	70.3%	97.7%	76%	66.8%	70%	97.6%	75.4%	65%	69%	97.4%	75.9%
MF	64.7%	66.7%	98.3%	70.2%	62.5%	65.3%	98.1%	70.1%	63.4%	65.7%	98.0%	70.2%
ARIMA	56.9%	46.8%	97.2%	55.8%	55.9%	48%	97.4%	56.1%	55.1%	47.9%	97.4%	54%
SARIMA	55.7%	42.7%	96.4%	38.8%	56.1%	45.8%	96.9%	33.5%	57%	47.8%	97%	39.9%
MICE	55.9%	56%	98.1%	61%	52.7%	54.5%	98.1%	60.4%	50.5%	53.1%	97.9%	58.3%

results listed in Fig. 9. All the methods were evaluated under the three cases defined in Table II. These benchmark methods require different input formats. For ARIMA and SARIMA, seven historical data points are used for inputs. In addition, the order of ARIMA and SARIMA are estimated by using a grid search procedure. EM, MF, and MICE impute missing data by using all the remaining data sets. The parameters of our model are listed in Section IV-C.

It is clear that the SSIM offers the best performance for the four metrics under all the three cases. For instance, the SSIM achieves 2.87, 3.11, and 3.29 RMSE scores in cases 1–3, respectively, as can be observed from Fig. 9(a). On the contrary, the best benchmark method MICE can only achieve 6.51, 6.57, and 6.64 for the three cases. Similarly, MICE also achieves the best MAE scores among all the comparative methods in all the three cases. However, MICE's MAE scores are still much worse than those of the SSIM [Fig. 9(b)].

The SSIM exhibits a remarkable performance when evaluated in the MAPE. The SSIM achieves 19.25%, 20.74%, and 23.45% MAPE scores in the three cases [Fig. 9(c)]. The best benchmark method is SARIMA in this evaluation but its performance is at least 27 times worse than ours. This is because the MAPE metric is sensitive when the ground truth values are near zero. When this occurs, biased predictions can result in very large MAPE value. In our experimental data, the concentration of nitrogen fluctuates around 0.15 mg/L in some testing time indexes, and can be as low as 0.1 mg/L (Table I). This also indicates that our model is able to achieve good imputation results, even when the nitrogen concentration is low. This is very important for the practical environmental monitoring systems because the concentration of nitrogen is usually maintained at a low level when there was less effects from irrigation or fertilization.

The SMAPE metric is also used in evaluating all the six comparative methods [Fig. 9(d)]. The way the SMAPE metric is defined helps diminish reduces the influence of near-zero values. However, the SSIM still achieves the best SMAPE scores for all the three cases. The SSIM has achieved 17.11%, 17.66%, and 19.01% SMAPE scores in the three cases, respectively. In this evaluation, SARIMA is also the best benchmark method but its performance was still worse than that of ours.

Table V summarizes the performance improvements of the proposed SSIM as opposed to the other five benchmark schemes in the three experimental cases. In this table, the improvement for the RMSE is calculated as $\Delta\text{Improvement} = ((\text{RMSE}_{\text{base}} - \text{RMSE}_{\text{ssim}})/\text{RMSE}_{\text{base}}) \times 100(\%)$, where $\text{RMSE}_{\text{base}}$ is the RMSE for the benchmark method, while $\text{RMSE}_{\text{ssim}}$ is the RMSE of the SSIM. The performance improvements are calculated in a similar manner for MAE, MAPE, and SMAPE.

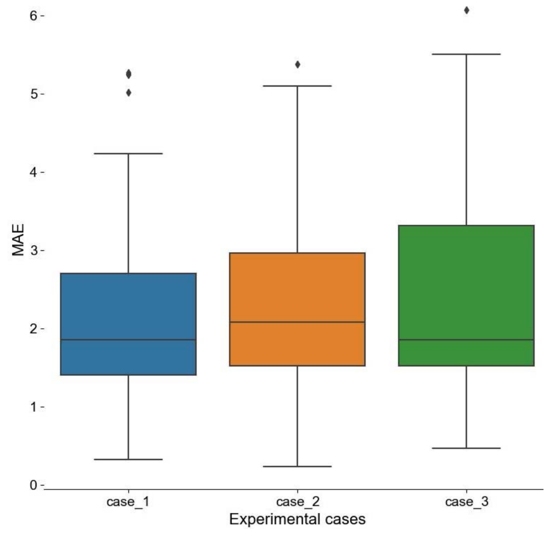


Fig. 10. Box-whisker plot for the MAE derived from the SSIM in three experimental cases.

In comparison with the other five comparative data imputation methods, the SSIM is able to improve the performance of all the metrics for the three cases, as can be observed from Table V. Our proposed method can achieve up to 69.2% improvement on the RMSE, up to 70.3% improvement on the MAE, up to 98.3% improvement on the MAPE, and up to 76% improvement on the SMAPE. It is evident that the SSIM offers the best performance among all the comparative methods.

Aside from the overall performance, it is also desirable for the data imputation methods to provide stable results. Fig. 10 depicts the distribution of the MAE of the SSIM through the quartiles. In all experimental cases, about 25% of imputed data pointed yielded by the SSIM have an MAE less than 2. Moreover, over 75% of imputed data points have an MAE less than 3 for the first and second cases. When imputing longer missing time series data in the third case, 75% of the results generated by the SSIM are still have an MAE less than 3.2. These results clearly demonstrate that the proposed SSIM is capable of imputing missing time series data with satisfactory accuracy for most testing samples.

V. CONCLUSION

Missing data is a common problem in sensor networks. Most data analytical methods require complete time series sensory data as inputs, and incomplete data can produce biased results.

In this paper, a deep learning-based data imputation model was proposed for recovering missing data sequences in sensor networks. Our model is designed to impute variable-length of missing data sequences by utilizing the information from both past and future time indexes. Moreover, a variable-length sliding window algorithm was developed to enhance the process of training sample generation. It allows one to generate large numbers of training samples based upon relatively small data sets, which is important for training deep learning models.

The experimental results were presented to demonstrate that the proposed model can recover missing data sequences more accurately than other benchmark methods, such as EM, MF, ARIMA, SARIMA, and MICE. Our model was shown to achieve the best RMSE, MAE, MAPE, and SMAPE scores in all the three experimental cases. Among the adopted performance metrics, our model produced outstanding MAPE scores, which demonstrates the excellent capability of our model in recovering missing time series data with low values. This property is significant in recovering environmental data because natural environment usually contains low level of some parameters like nitrogen. Furthermore, our model was able to provide stable imputation results in experiments we performed, suggesting the proposed approach is very promising for data recovery problems in sensor networks.

In future work, we intend to make use of sensory data from other monitoring stations to pretrain the deep learning model. Also, spatial information will be investigated and integrated into our model with the objective of further improving the data imputation accuracy. Furthermore, we will improve our deep learning model with other architectures such as the convolutional neural network to better exploit the long term temporal dependencies in time-series sensor data.

Acknowledgment

The authors would like to thank the Great Barrier Reef Catchment Loads Monitoring Program for providing valuable real-time water quality monitoring data sets.

References

[1] M. Rode *et al.*, "Sensors in the stream: The high-frequency wave of the present," *Environ. Sci. Technol.*, vol. 50, no. 19, pp. 10297–10307, Oct. 2016.

[2] J. W. Graham, "Missing data analysis: Making it work in the real world," *Annu. Rev. Psychol.*, vol. 60, pp. 549–576, Jan. 2009.

[3] B. Fekade, T. Maksymyuk, M. Kyryk, and M. Jo, "Probabilistic recovery of incomplete sensed data in IoT," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2282–2292, Aug. 2018.

[4] J. Zhou and Z. Huang, "Recover missing sensor data with iterative imputing network," in *Proc. Workshops 32nd Artif. Intell. (AAAI) Conf.*, New Orleans, LA, USA, Feb. 2018, pp. 209–215.

[5] C. Leke, B. Twala, and T. Marwala, "Missing data prediction and classification: The use of auto-associative neural networks and optimization algorithms," *arXiv preprint arXiv:1403.5488*, 2014.

[6] *Center for Machine Learning and Intelligent Systems*. Accessed: Jul. 20, 2018. [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html

[7] N. Jaques, S. Taylor, A. Sano, and R. Picard, "Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction," in *Proc. 7th Int. Conf. Affective Comput. Intell. Interact. (ACII)*, San Antonio, TX, USA, Oct. 2017, pp. 202–208.

[8] M. Das and S. K. Ghosh, "A deep-learning-based forecasting ensemble to predict missing data for remote sensing analysis," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 12, pp. 5228–5236, Dec. 2017.

[9] 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 *Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput.*, 2018, pp. 1293–1300.

[10] H. Verma and S. Kumar, "An accurate missing data prediction method using LSTM based deep learning for health care," in *Proc. 20th Int. Conf. Distrib. Comput. Netw.*, 2019, pp. 371–376.

[11] L. Shen, Q. Ma, and S. Li, "End-to-end time series imputation via residual short paths," in *Proc. Asian Conf. Mach. Learn.*, 2018, pp. 248–263.

[12] L. Li, J. Zhang, Y. Wang, and B. Ran, "Missing value imputation for traffic-related time series data based on a multi-view learning method," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[13] Z. C. Lipton, D. C. Kale, and R. Wetzel, "Modeling missing data in clinical time series with RNNS," in *Proc. Mach. Learn. Healthcare*, 2016, pp. 1–17.

[14] 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.*, to be published.

[15] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Empirical Methods Nat. Lang. Process. (EMNLP) Conf.*, Doha, Qatar, Oct. 2014, pp. 1724–1734.

[16] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 3156–3164.

[17] O. Vinyals *et al.*, "Grammar as a foreign language," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2. Montreal, QC, Canada, Dec. 2015, pp. 2773–2781.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[19] R. J. Little and D. B. Rubin, *Statistical Analysis With Missing Data*. Hoboken, NJ, USA: Wiley, 2014.

[20] A. Farhangfar, L. Kurgan, and J. Dy, "Impact of imputation of missing values on classification error for discrete data," *Pattern Recognit.*, vol. 41, no. 12, pp. 3692–3705, Dec. 2008.

[21] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[22] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, Lisbon, Portugal, Sep. 2015, pp. 1412–1421.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[24] *Reef 2050 Water Quality Improvement Plan*. Accessed: Jul. 20, 2018. [Online]. Available: https://www.reefplan.qld.gov.au/

[25] *Bureau of Meteorology Weather Stations*. Accessed: Jul. 20, 2018. [Online]. Available: http://www.bom.gov.au/climate/data/stations/

[26] L. Fu and Y.-G. Wang, "Statistical tools for analyzing water quality data," in *Water Quality Monitoring and Assessment*. London, U.K.: IntechOpen, 2012.

[27] R. J. Sänket, A. Baviskar, S. S. Ahire, and S. V. Mapare, "Development of a low cost nitrate detection soil sensor," in *Proc. Int. Conf. Wireless Commun. Signal Process. Netw. (WiSPNET)*, Chennai, India, Mar. 2017, pp. 1272–1275.

[28] S. Makridakis, "Accuracy measures: Theoretical and practical concerns," *Int. J. Forecasting*, vol. 9, no. 4, pp. 527–529, 1993.

[29] C. Chen, J. Twycross, and J. M. Garibaldi, "A new accuracy measure based on bounded relative error for time series forecasting," *PLoS ONE*, vol. 12, no. 3, 2017, Art. no. e0174202.

[30] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

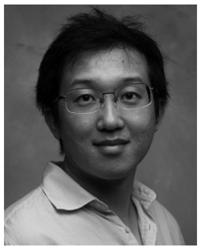
[31] N. G. Reich *et al.*, "Case study in evaluating time series prediction models using the relative mean absolute error," *Amer. Stat.*, vol. 70, no. 3, pp. 285–292, 2016.

[32] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden-Day, 1990.

[33] A. Pankratz, *Forecasting With Univariate Box—Jenkins Models: Concepts and Cases* (Wiley Series in Probability and Statistics). New York, NY, USA: Wiley, 2009.

[34] A. M. Buchanan and A. W. Fitzgibbon, "Damped Newton algorithms for matrix factorization with missing data," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. San Diego, CA, USA, Jun. 2005, pp. 316–322.

[35] M. Azur, E. Stuart, C. Frangakis, and P. Leaf, "Multiple imputation by chained equations: What is it and how does it work?" *Int. J. Methods Psychiatric Res.*, vol. 20, no. 1, pp. 40–49, Mar. 2011.
[36] H. M. Ghomrawi, L. A. Mandl, J. Rutledge, M. M. Alexiades, and M. Mazumdar, "Is there a role for expectation maximization imputation in addressing missing data in research using WOMAC questionnaire? Comparison to the standard mean approach and a tutorial," *BMC Musculoskeletal Disorders*, vol. 12, no. 1, p. 109, May 2011.
[37] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://keras.io



Yi-Fan Zhang (S'08–M'13) received the B.Eng. and M.Eng. degrees in reliability and systems engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2008 and 2011, respectively, and the Ph.D. degree in data science from the Queensland University of Technology, Brisbane, QLD, Australia, in 2016.

He is currently a Post-Doctoral Fellow with the Department of Agriculture and Food, CSIRO, Brisbane, QLD, Australia. His current research interests include artificial intelligence, time series modeling, cloud computing, and Internet of Things.

Peter J. Thorburn is an Agricultural Scientist with strong multidisciplinary interests in the dynamics of soil-plant interactions and a strong commitment to enhancing the sustainability of agricultural systems. With a background in soil science and plant physiology, his research focuses on developing and applying simulation models to understand soil and plant interactions in agricultural production systems, aiming to determine management systems that can reduce detrimental environmental impacts while, still continuing to produce significant economic and social outcomes in current and future climates. He is a Senior Principal Research Scientist with the Department of Agriculture and Food, CSIRO, Brisbane, QLD, Australia. He is the CSIRO representative on the APSIM Imitative, the joint venture that owns the APSIM farming systems model. He also coordinates research on agriculture's effect on the Great Barrier Reef across CSIRO's Agriculture and Food Department and co-leads the strategic collaboration between CSIRO's Agriculture and Food Department and AgResearch, Ruakura, New Zealand. His international activities include co-leading, with Prof. Boote from the University of Florida, the Crop Modelling stream of the Agricultural Modeling Improvement and Intercomparison Program (AgMIP), an international collaboration with American and European institutions developing increased capacity for quantitatively assessing climate change impacts on global food security. He is also on the Science Advisory Group of New Zealand's OVERSEER agricultural nutrient management model.

Wei Xiang (S'00–M'04–SM'10) received the B.Eng. and M.Eng. degrees in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1997 and 2000, respectively, and the Ph.D. degree in telecommunications engineering from the University of South Australia, Adelaide, SA, Australia, in 2004.

From 2004 to 2015, he was with the School of Mechanical and Electrical Engineering, University of Southern Queensland, Toowoomba, QLD, Australia. He is currently the Founding Professor and the Head of the Discipline of Internet of Things Engineering, James Cook University, Cairns, QLD, Australia. He has authored or co-authored over 250 peer-reviewed papers, including 3 academic books and 130 journal papers. His current research interests include communications and information theory, particularly the Internet of Things, and coding and signal processing for multimedia communications systems.

Dr. Xiang was a recipient of the TNQ Innovation Award in 2016, the Pearcey Entrepreneurship Award in 2017, the Engineers Australia Cairns Engineer of the Year in 2017, the Queensland International Fellow by the Queensland Government of Australia, from 2010 to 2011, the Endeavour Research Fellow by the Commonwealth Government of Australia, from 2012 to 2013, the Smart Futures Fellow by the Queensland Government of Australia, from 2012 to 2015, and the JSPS Invitational Fellow jointly by the Australian Academy of Science and the Japanese Society for Promotion of Science, from 2014 to 2015, and several prestigious fellowship titles. He was a co-recipient of four Best Paper Awards of WiSATS 2019, WCSP 2015, the IEEE WCNC 2011, and ICWMC 2009. Due to his instrumental leadership in establishing Australia's first accredited Internet of Things Engineering degree program, he was selected into Pearcy Foundation's Hall of Fame in 2018. He is an elected Fellow of the IET in U.K. and Engineers Australia. He is the Vice Chair of the IEEE Northern Australia Section. He was an Editor of IEEE COMMUNICATIONS LETTERS from 2015 to 2017. He is an Associate Editor of Springer's *Telecommunications Systems*. He has served on a large number of international conferences as the General Co-Chair, the TPC Co-Chair, and the Symposium Chair.

Peter Fitch is a Principal Research Scientist with the Department of Land and Water, CSIRO, Canberra, ACT, Australia. He is a Research Leader and a Manager with over 20 years research and industry leadership and management experience. He has led the CSIRO environmental information systems research program in CSIRO's Land and Water Department for five years until 2014 and currently leads projects as part of the Digiscape Future Science Platform initiative. He has significant experience in program development and management, development of research strategy, and implementation of research plans.