NP-Free: A Real-Time Normalization-free and Parameter-tuning-free Representation Approach for Open-ended Time Series

Ming-Chang Lee
Dept. of Computer Science
Høgskulen på Vestlandet (HVL)
Bergen, Norway
mingchang1109@gmail.com
ming-chang.lee@hvl.no

Jia-Chun Lin
Dept. of Information Security and Communication Technology
Norwegian University of Science and Technology (NTNU)
Gjøvik, Norway
jia-chun.lin@ntnu.no

Volker Stolz
Dept. of Computer Science
Høgskulen på Vestlandet (HVL)
Bergen, Norway
volker.stolz@hvl.no

***Abstract*—To help analyze time series in data mining applications, many time series representation approaches have been proposed to convert a raw time series into another series for representing the original time series. However, existing approaches are not designed for open-ended time series (which is a sequence of data points being continuously collected at a fixed interval without any length limit) because these approaches need to know the total length of the target time series in advance and preprocess the entire time series using a normalization method. Furthermore, many representation approaches require users to configure and tune some parameters beforehand in order to achieve satisfactory representation results. In this paper, we propose NP-Free, a real-time Normalization-free and Parameter-tuning-free representation approach for open-ended time series. Without needing to use any normalization method or tune any parameter, NP-Free can generate a representation for a raw time series on the fly by converting each data point of the time series into a root-mean-square error (RMSE) value based on Long Short-Term Memory (LSTM) and a Look-Back and Predict-Forward strategy. To demonstrate the capability of NP-Free in representing time series, we conducted several experiments based on real-world open-source time series datasets. We also evaluated the time consumption of NP-Free in generating representations.**

***Index Terms*—open-ended time series, time series representation, dimensionality reduction, parameter-tuning-free, z-normalization, time series analysis, data mining**

I. Introduction

In recent years, there has been an increasing need for time series analysis such as clustering, classification, anomaly detection, forecasting, indexing, etc. [2] [5] [6] [18] [25] [26] due to the explosion of the Internet of Things in a cyber-physical world. Large volumes of time series data are continuously measured and collected from connected devices and sensors. Analyzing raw time series can be challenging and undesirable due to high computation cost and memory requirement [13]. One solution to achieve effective and efficient time series analysis is via high-level representation approaches, which aim to extract features of time series or reduce the dimensionality of the time series while preserving the fundamental characteristics of the time series [16].

A number of time series representation approaches have been proposed in the last few years. However, many of them work only on fixed-length time series, rather than on open-ended time series. This is something to do with their underlying designs. Before generating a representation for a time series, these approaches need to preprocess the time series using z-normalization (also known as z-score normalization), a commonly used normalization approach for time series [14]. However, z-normalization might cause two distinct time series to become indistinguishable [7], therefore misleading the representation approaches and negatively affecting subsequent data mining tasks. Another shortcoming with most representation approaches is that users require to configure certain parameters (e.g., the length of the time series, the size of sliding window, or the size of Alphabet, etc. [8]) in advance. Without a proper value to each parameter, these approaches may generate improper representations for time series, consequently affecting subsequent data mining tasks.

To address aforementioned issues, this paper introduces NP-Free, which is a real-time Normalization-free and Parameter-tuning-free representation approach for open-ended time series. Inspired by one of the state-of-the-art real-time time series anomaly detection approaches called RePAD [2], NP-Free adopts Long Short-Term Memory (LSTM) and the Look-Back and Predict-Forward strategy used by RePAD to generate representations for time series. Note that NP-Free does not need to preprocess time series using any normalization approach or require users to tune any parameter. In fact, NP-Free can directly generate a representation for a raw time series on the fly by transforming the time series into a root-mean-square error (RMSE) series in real time (i.e., instantaneously).

More specifically, NP-Free keeps predicting the next data point of the target time series based on three historical data points and keeps calculating the corresponding RMSE value between the observed and predicted data points. By doing so, NP-Free can convert the target time series into a RMSE series. To adapt to any pattern change in the time series, NP-Free also retrains a new LSTM model as needed based on the self-adaptive detection threshold introduced in [2].

NP-Free can be applied to any open-ended univariate time series regardless of the pattern (i.e., recurrent or non-recurrent) since NP-Free does not require the knowledge of the total length of the target time series in advance. To demonstrate

the capability of NP-Free in representing time series, we performed experiments based on real-world open-source time series datasets. The results demonstrate that the representations generated by NP-Free can retain the fundamental structural properties of the corresponding time series and can therefore represent the time series.

The rest of the paper is organized as follows: Sections II and III present z-normalization and related work, respectively. In Section IV, we introduce NP-Free. Section V presents and discusses the experiments and the corresponding results. Section VI concludes this paper and outlines future work.

II. Z-NORMALIZATION

Z-normalization, also known as z-score normalization, is the process of normalizing each value in a dataset such that the mean of all values is approximately 0 and the standard deviation is in a range close to 1 [20]. The formula of z-normalization is shown below.

$$\overline{x}_i = \frac{x_i - \mu}{\sigma} \tag{1}$$

In the context of time series, x_i is the i-th data point of the time series, μ is the mean of all data points in the time series, σ is the standard deviation of all the data points, and $\overline{x}_i$ is normalized data point derived from the equation. Z-normalization is considered an essential preprocessing step which allows time series representation approaches to focus on the structural similarities/dissimilarities of time series rather than on the original data point values [20]. However, z-normalization has some shortcomings. When it encounters a flat time series, any fluctuations (e.g., noises) will be enhanced, resulting in an negative impact to a data mining technique, such as Matrix Profile [17]. According to [7], z-normalization might also destroy potentially relevant properties to distinguish time series since the time series are shifted to have zero mean. In other words, two distinct time series might become indistinguishable after z-normalization. Similarly, according to [14] [20] [21], z-normalization might not produce normalized data with the exact same scale, causing a negative impact to subsequent data mining tasks. This phenomenon is also shown in our preprint paper[1].

III. RELATED WORK

As mentioned earlier, time series representations are often treated as the first component of time series analysis [16]. According to [13], time series representation approaches can be classified into two categories: data adaptive and non-data adaptive. Data adaptive representation methods, including Symbolic Aggregate approXimation (SAX) [8] and the clipped representation approach [19], try to minimize the global reconstruction error using arbitrary length segments [16]. According to [16], this type of methods can better approximate each time series in datasets, but the comparison of several time series is difficult. In contrast, non-data adaptive approaches, including Piecewise Aggregate Approxmiation (PAA) [12] and Discrete Fourier Transformation (DFT) [32], considered local properties of time series and constructed an approximate representation accordingly [28]. Hence, non-data adaptive approaches are suitable for time-series with equal-length segmentation, and it is straightforward to compare the representations of several time series [16].

Despite the above classification, existing representation approaches are designed to generate representations for fixed-length time series because they require to normalize the target time series using z-normalization before generating a representation for the time series. Example approaches include SAX [8], PAA [12], Symbolic Aggregate approXimation and Vector Space Model (SAX-VSM) [15], the clipped representation approach [19], etc. Furthermore, they also require users to set some parameters in advance. Without a proper value to each parameter, these approaches might not be able to generate proper representations for time series.

IV. METHODOLOGY OF NP-FREE

NP-Free aims to convert an open-ended univariate time series into a RMSE series in real time for representing the time series. To achieve this goal, NP-Free adopts some strategies used by a lightweight real-time time series anomaly detection approach called RePAD [2], including the simple LSTM network structure (i.e., one hidden layer with ten hidden units), the Look-Back and Predict-Forward strategy, and the main idea of the detection threshold. In addition, NP-Free follows the suggestion from [4] to set the Look-Back parameter to 3. In other words, NP-Free always uses three historical data points to predict each upcoming data point.

However, unlike RePAD, NP-Free is not designed to detect anomalies. Instead, it is to predict every upcoming data point in the target time series and calculate the corresponding RMSE value. To make sure the deterministic property of NP-Free (i.e., the same RMSE series will always be generated given the same time series), NP-Free removes all randomness by fixing all LSTM hyperparameter setting (including the number of hidden layers, the number of hidden units, learning rate, the number of epochs, etc.) and following the hyperparameter setting used by RePAD [2].

Fig. 1 illustrates the algorithm of NP-Free where T denotes the current time point and T starts from 0, which is the time point when NP-Free is launched. Let d_T be the real data point collected at time point T, and $\widehat{d_T}$ be the data point predicted by NP-Free for time point T. There are four differences between NP-Free and RePAD. First, NP-Free always uses three data points to predict each upcoming data point. In other words, it removes the Look-Back parameter by fixing the value of it to three, followed by the suggestion made by [4].

The second difference is that NP-Free uses RMSE to measure its prediction error instead of using AARE. Equation 6 shows how $RMSE_T$ (i.e., the RMSE at time point T) is calculated.

$$RMSE_T = \sqrt{\frac{\sum_{z=T-2}^{T} (d_z - \widehat{d_z})^2}{3}}, T \geq 5 \tag{2}$$

[1]NP-Free preprint, https://doi.org/10.48550/arXiv.2304.06168

It is clear that given any two time series (says *A* and *B*), RMSE is to evaluate the absolute error between *A* and *B*, meaning that the error would be always identical no matter *A* is the observed time series or the predicted one. On the contrary, AARE is to evaluate the relative error. Hence, the relative error of *A* to *B* might not be the same as the relative error of *B* to *A*. Due to the above reason, NP-Free chose RMSE to help achieve the above-mentioned deterministic property.

NP-Free algorithm
Input: Each data point of the target time series
Output: A RMSE value
Procedure:

1: Let T be the current time point and T starts from 0; Let *Flag* be True;
2: **While** time has advanced {
3: Collect data point d_T;
4: **if** $T \geq 2$ and $T < 5$ {
5: Train an LSTM model by taking d_{T-2}, d_{T-1}, and d_T as the training data;
6: Let m be the resulting LSTM model and use m to predict $\widehat{d_{T+1}}$;}
7: **else if** $T \geq 5$ and $T < 7$ {
8: Calculate $RMSE_T$ based on Equation 2 and output $RMSE_T$;
9: Train an LSTM model by taking d_{T-2}, d_{T-1}, and d_T as the training data;
10: Let m be the resulting LSTM model and use m to predict $\widehat{d_{T+1}}$;}
11: **else if** $T \geq 7$ and *Flag*==True {
12: **if** $T \neq 7$ { Use m to predict $\widehat{d_T}$;}
13: Calculate $RMSE_T$ based on Equation 2;
14: Calculate thd_{RMSE} based on Equation 3;
15: **if** $RMSE_T \leq thd_{RMSE}$ { Output $RMSE_T$;}
16: **else**{
17: Train an LSTM model with d_{T-3}, d_{T-2}, and d_{T-1};
18: Use the newly trained LSTM model to predict $\widehat{d_T}$;
19: Calculate $RMSE_T$ based on Equation 2;
20: Calculate thd_{RMSE} based on Equation 3;
21: **if** $RMSE_T \leq thd_{RMSE}$ { Output $RMSE_T$;}
22: **else** { Output $RMSE_T$; Let *Flag* be False;}}}
23: **else if** $T \geq 7$ and *Flag*==False {
24: Train an LSTM model with d_{T-3}, d_{T-2}, and d_{T-1};
25: Use the newly trained LSTM model to predict $\widehat{d_T}$;
26: Calculate $RMSE_T$ based on Equation 2;
27: Calculate thd_{RMSE} based on Equation 3;
28: **if** $RMSE_T \leq thd_{RMSE}$ {
29: Output $RMSE_T$;
30: Replace m with the new LSTM model from line 24;
31: Let *Flag* be True;}
32: **else** { Output $RMSE_T$; Let *Flag* be False;}}}

Fig. 1. The algorithm of NP-Free.

The third difference is that NP-Free keeps calculating its threshold thd_{RMSE} (see Equation 3) based on a fixed number of RMSE values, rather than all past AARE values.

$$thd_{RMSE} = M_{RMSE} + 3 \cdot \sigma \tag{3}$$

M_{RMSE} is the average RMSE at time point T as shown below.

$$M_{RMSE} = \begin{cases} \frac{1}{T-4} \cdot \sum_{z=5}^{T} RMSE_z, 7 \leq T < W+4 \\ \frac{1}{W} \cdot \sum_{z=T-W+1}^{T} RMSE_z, T \geq W+4 \end{cases} \tag{4}$$

Variable W is the length of a sliding window to control how many previous RMSE values should be considered to derive the threshold. For instance, when W is set to 100, NP-Free will always use the 100 most recently derived RMSE values to calculate M_{RMSE} if the total number of historical RMSE values is sufficient. Since NP-Free is designed to generate representations for open-ended time series, it is important to include a limited sliding window so that the underlying system resources will not be exhausted by Equation 3.

Equation 5 shows how to calculate standard deviation σ at time point T. Similarly, sliding window W is also considered to derive σ.

$$\sigma = \begin{cases} \sqrt{\frac{\sum_{z=5}^{T} (RMSE_z - M_{RMSE})^2}{T-4}}, 7 \leq T < W+4 \\ \sqrt{\frac{\sum_{z=T-W+1}^{T} (RMSE_z - M_{RMSE})^2}{W}}, T \geq W+4 \end{cases} \tag{5}$$

Whenever time point T advances and it is greater than or equal to 7 (i.e., either line 11 of Fig. 1 or line 23 of Fig. 1 is evaluated to be true), NP-Free re-calculates $RMSE_T$ and thd_{RMSE} . If $RMSE_T$ is not greater than the threshold (see lines 15 and 28), $RMSE_T$ will be immediately outputted by NP-Free.

Otherwise, it might mean that the data pattern of the target time series has changed. In this case, NP-Free will try to adapt to the change by retraining an new LSTM model to re-predict $\widehat{d_T}$ and re-calculate both $RMSE_T$ and thd_{RMSE} either at current time point (i.e., lines 17 to 20) or at the next time point (i.e., lines 24 to 27) by setting its *Flag* to be "False". If the re-calculated $RMSE_T$ is not larger than thd_{RMSE}, NP-Free immediately outputs $RMSE_T$. Otherwise, NP-Free outputs $RMSE_T$ and sets its *Flag* to be "False", which will trigger LSTM model retraining at the next time point.

The last difference between NP-Free and RePAD is that NP-Free does not announce if a data point is anomalous or not because detecting anomalies is not its goal. By repeating the same process, an open-ended time series can be converted into a RMSE series on the fly in real time. In the next section, we will demonstrate the RMSE series generated by NP-Free can indeed represent the corresponding time series.

V. Experimental Results

We conducted a set of experiments to demonstrate the capability of NP-Free in generating time series representations. Note that we did not compare NP-Free with other representation approaches because none of them was designed to generate representations for open-ended time series in real time. In other words, we did not use *TLB* (Tightness of Lower Bound) [13] to evaluate NP-Free even though *TLB* is a common measure for comparing representation approaches. According to the equation of *TLB*, i.e., *TLB=LowerBoundDist(X,Y)/TrueEuclideanDist(X,Y)*, where *X* and *Y* are two z-normalized time series. Since NP-Free works directly on raw time series, rather than on z-normalized time series, *TLB* cannot be applied to evaluate NP-Free.

TABLE I
The hyperparameter and parameter setting used by NP-Free.

Hyperparameters and parameters	Value
The number of hidden layers	1
The number of hidden units	10
The number of epochs	50
Learning rate	0.005
Activation function	tanh
Random seed	140
The sliding window	1440

TABLE I lists all hyperparameters and parameters used and pre-fixed by NP-Free. Note that all of them (except the sliding window) were used or suggested by [2] and [4]. Regarding the sliding window, we conducted an experiment to study how different sliding window sizes impact NP-Free. We chose 288, 576, 864, 1152, 1440, 1728, 2016, and 4032 to be the sliding window, but found that all of them had no significant impact to NP-Free in terms of representation generation and time consumption. Hence, we chose one of them (1440) to be the sliding window in our experiments. In other words, users do not need to tune any hyperparameters or parameters for

using NP-Free. That is why we said that NP-Free is parameter-tuning-free. The main purpose of fixing all hyperparameters and parameters is to ensure that NP-Free can always produce the same RMSE series for the same time series (i.e., the deterministic property).

In this paper, NP-Free was implemented in Deeplearning4j [37], and all experiments were conducted on a MacBookPro laptop running MacOS Monterey 12.5.1 with Apple M1 Pro chip and 16GB Memory. The purpose is to demonstrate that NP-Free can be easily deployed on a commodity machine.

We chose three time series from an open-source repository called NAB [3] and one time series from Melbourne Pedestrian Foot Traffic [33]. TABLE II lists the details of the four time series. The first two time series are separately called rds-cpu-utilization-e47b3b (B3B for short) and ec2-cpu-utilization-825cc2 (CC2 for short), and both exhibit a non-recurrent data pattern. The last two time series are separately called art-daily-small-noise (ADSN for short) and Bourke Street Mall South (BSMS for short), and they exhibit a recurrent data pattern. The reason of choosing these time series is to show that NP-Free works on any time series regardless of their data patterns.

TABLE II
FOUR OPEN-SOURCE TIME SERIES USED.

Name	Time Period	Total number of data points	Time Interval
B3B	From 2014-04-10, 00:02 to 2014-04-23, 23:57	4032	5 min
CC2	From 2014-04-10, 00:04 to 2014-04-24, 00:09	4032	5 min
ADSN	From 2014-04-01, 00:00 to 2014-04-14, 23:55	4032	5 min
BSMS	From 2016-01-01, 00:00 to 2016-06-30, 23:00	4368	1 hour

To show that NP-Free always generates similar RMSE series for similar *non-recurrent* time series, we created 12 variants for time series B3B and named them B3B+100, B3B+200, ..., B3B+1000, B3B+1500, and B3B+2000 by increasing each data point value of the B3B time series by 100, 200, ..., 1000, 1500, and 2000, respectively. Due to page limit, we only show B3B and B3B+100 in Fig. 2. It is clear that these two time series have the exactly the same shape and data pattern but have different offsets in the Y axis. In fact, all the B3B variants have the same shape and data pattern as B3B.

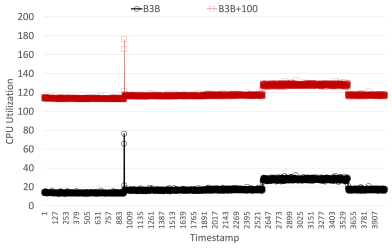

Fig. 2. All data points in B3B and B3B+100.

Then we used NP-Free to generate a RMSE series for B3B and each B3B variant. Just for illustration, Fig. 3 depicts the RMSE series generated by NP-Free for B3B and B3B+100. Apparently, it is difficult to distinguish the two RMSE series with human eyes because they almost overlap with each other. Therefore, we utilized Euclidean Distance [11] (see Equation 6) to compare the two RMSE series.

$$ED(R_i,R_j) = \sqrt{\sum_{z=1}^{L} (R_{i,z} - R_{j,z})^2} \quad (6)$$

where R_i and R_j are any two RMSE series (i=1, 2,..., n, and j=1, 2,..., n, but $i \neq j$), $R_{i,z}$ denotes the z-th data point of R_i, and $R_{j,z}$ denotes the z-th data point of R_j, where z=1, 2, ..., L. We calculated the Euclidean Distance between the RMSE series of B3B and the RMSE series of every other B3B variant separately. As listed in TABLE III, all Euclidean Distance values are very low, meaning that the RMSE series generated by NP-Free for all the B3B variants are individually similar to the RMSE series generated by NP-Free for B3B.

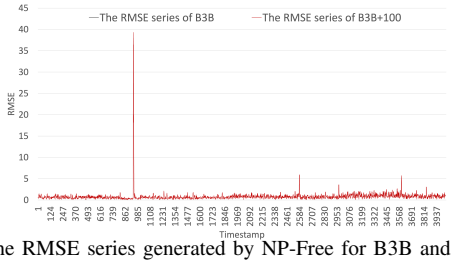

Fig. 3. The RMSE series generated by NP-Free for B3B and B3B+100.

TABLE III
THE EUCLIDEAN DISTANCE BETWEEN THE RMSE SERIES OF EACH B3B VARIANT AND THE RMSE SERIES OF B3B.

Variant Name	Euclidean Distance to B3B
B3B+100	0.060
B3B+200	0.62
B3B+300, B3B+400	0.060
B3B+500, B3B+600, ..., B3B+900	0.62
B3B+1000	0.216
B3B+1500, B3B+2000	0.213

TABLE IV
THE EUCLIDEAN DISTANCE BETWEEN THE RMSE SERIES OF EACH CC2 VARIANT AND THE RMSE SERIES OF CC2.

Variant Name	Euclidean Distance to CC2
CC2+100	0.007
CC2+200, CC2+300, CC2+400	0.016
CC2+500, CC2+600, ..., CC2+900	0.017
CC2+1000, CC2+1500	0.23
CC2+2000	0.227

We repeated the same procedure to create 12 variants for CC2, and used NP-Free to generate a RMSE series for CC2 and each CC2 variant. TABLE IV shows that the Euclidean Distance between the RMSE series of each CC2 variant and that of CC2 is very short, i.e., NP-Free indeed generates similar RMSE series for similar non-recurrent time series.

Now we continue to demonstrate that NP-Free can generate similar RMSE series for similar *recurrent* time series. To do it, we followed the same procedure to separately create 12 variants for ADSN and BSMS so that all the variants for ADSN have the same data pattern as ADSN, and that all the variants for BSMS have the same data pattern as BSMS.

Due to the page limit, we only show ADSN and one of its variants (called ADSN+100) in Fig. 4. Apparently,

they have exactly the same shape and data pattern. Fig. 5 depicts the RMSE series generated by NP-Free for ADSN and ADSN+100. Clearly, it is hard to distinguish the two RMSE series due to the significant overlapping. Hence, we further calculated the Euclidean Distance between the RMSE series of ADSN and the RMSE series of each ADSN variant. As shown in TABLE V, all distance values are very low.

We also obtained similar results when NP-Free was used to generate RMSE series for BSMS and each BSMS variant. As listed in TABLE VI, all Euclidean Distance values are very low as well. Based on the above results, we confirm that NP-Free generates similar RMSE series for similar time series. The results for demonstrating that NP-Free does not generate similar RMSE series when time series have an opposite data pattern can be found in our preprint[2].

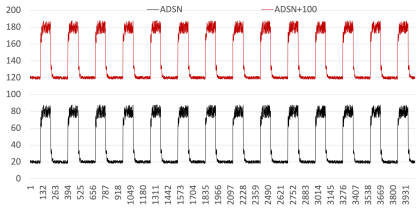

Fig. 4. All data points in ADSN and ADSN+100.

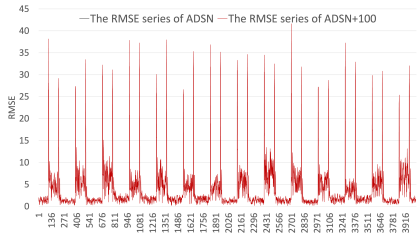

Fig. 5. The RMSE series generated by NP-Free for ADSN and ADSN+100.

TABLE V
THE EUCLIDEAN DISTANCE BETWEEN THE RMSE SERIES OF EACH ADSN VARIANT AND THE RMSE SERIES OF ADSN.

Variant Name	Euclidean Distance to ADSN
ADSN+100, ADSN+200, ..., ADSN+400	0.08
ADSN+500, ADSN+600, ..., ADSN+900	0.083
ADSN+1000, ADSN+1500	0.247
ADSN+2000	0.246

TABLE VI
THE EUCLIDEAN DISTANCE BETWEEN THE RMSE SERIES OF EACH BSMS VARIANT AND THE RMSE SERIES OF BSMS.

Variant Name	Euclidean Distance to BSMS
BSMS+100	0.005
BSMS+200	0.008
BSMS+300	0.011
BSMS+400	0.014
BSMS+500, BSMS+700, BSMS+700	0.016
BSMS+800	0.020
BSMS+900	0.022
BSMS+1000	0.023
BSMS+1500	0.027
BSMS+2000	0.035

Recall that NP-Free has a lightweight design. To demonstrate that NP-Free is indeed lightweight, we separately evaluated the time consumption of NP-Free in generating RMSE representations for time series B3B, CC2, ADSN, and BSMS. Recall that NP-Free only trains an LSTM model in the beginning and when it finds that current RMSE value is greater than the current value of thd_{RMSE}. TABLE VII shows the number of LSTM model training required for each time series. When NP-Free converted B3B into the corresponding RMSE time series, only 59 data points out of 4032 data points triggered LSTM model training, implying that the training ratio is 1.46% (=59/4032). It is also clear from TABLE VII that the training ratios for the other three time series are very low, implying that NP-Free did not need to train its LSTM models often.

TABLE VII
TOTAL NUMBER OF REQUIRED LSTM MODEL RETRAINING.

Time Series	Number of data points triggering LSTM model training
B3B	59 out of 4032
CC2	61 out of 4032
ADSN	127 out of 4032
BSNS	15 out of 4368

When LSTM model retraining is required, NP-Free might require more time to generate a RMSE value. TABLE VIII lists the average time required by NP-Free to generate a RMSE value for each data point in the four different time series while LSTM model retraining is required. On the other hand, TABLE IX shows the average time required by NP-Free while LSTM model training is not required. Apparently, when LSTM model retraining is required, NP-Free needs more time to generate a RMSE value. Nevertheless, we can see that NP-Free is very efficient since it does not need to retrain LSTM models very often and that even when it needs to do so, it can generate a RMSE value in a short time. That is why we said NP-Free can generate representations in real time.

TABLE VIII
TIME CONSUMPTION WHILE LSTM MODEL RETRAINING IS REQUIRED.

Time Series	Average time of generating a RMSE Value	Std dev
B3B	0.233 sec	0.045 sec
CC2	0.241 sec	0.061 sec
ADSN	0.217 sec	0.048 sec
BSNS	0.291 sec	0.073 sec

TABLE IX
TIME CONSUMPTION WHILE MODEL RETRAINING IS NOT REQUIRED.

Time Series	Average time of generating a RMSE Value	Std dev
B3B	0.007 sec	0.004 sec
CC2	0.008 sec	0.07 sec
ADSN	0.008 sec	0.003 sec
BSNS	0.007 sec	0.005 sec

VI. CONCLUSION AND FUTURE WORK

In this paper, we have introduced NP-Free. It can generate a representation for an open-ended univariate time series in real time by predicting each upcoming data point of the time series and calculating the corresponding RMSE value on the fly. Different from existing representation approaches, NP-Free does not need to go through any offline normalization preprocessing. Besides, it does not require users to tune or configure any hyperparameter or parameter.

[2]NP-Free preprint, https://doi.org/10.48550/arXiv.2304.06168

The experiment based on real-world open-source time series datasets demonstrated that the RMSE series generated by NP-Free can indeed represent the corresponding time series. The results show that NP-Free indeed generates similar RMSE series for similar time series regardless of their offsets in the Y axis. Furthermore, the experiments also show that NP-Free is very lightweight because it employs a simple LSTM network (only one hidden layer with ten hidden units), always uses three data points to train its LSTM model, and performs LSTM model retraining only when necessary. Therefore, NP-Free can be easily deployed on commodity computers and used as a pre-processing tool for real-time time series analysis, such as clustering and classification.

As for future work, we plan to design a real-time clustering algorithm for clustering time series based on NP-Free. Additionally, we intend to propose a real-time online pattern recognition approach based on NP-Free.

REFERENCES

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

[2] M.-C. Lee, J.-C. Lin, and E. G. Gran, "RePAD: real-time proactive anomaly detection for time series," In Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA 2020), pp. 1291–1302, 2020. (The updated version of the RePAD algorithm from arXiv was used in this NP-Free paper. See arXiv:2001.08922)

[3] NAB [Online code repository], Available: https://github.com/numenta/NAB [accessed 12-April-2023]

[4] M.-C. Lee, J.-C. Lin, and E. G. Gran, "How Far Should We Look Back to Achieve Effective Real-Time Time-Series Anomaly Detection?," International Conference on Advanced Information Networking and Applications (AINA 2021), pp. 136–148, 2021. arXiv preprint arXiv:2102.06560.

[5] M.-C. Lee, J.-C. Lin, and E. G. Gran, "ReRe: a lightweight real-time ready-to-go anomaly detection approach for time series," 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 322–327, 2020. arXiv preprint arXiv:2004.02319.

[6] M.-C. Lee, J.-C. Lin, and E. G. Gran, "SALAD: Self-adaptive lightweight anomaly detection for real-time recurrent time series," 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 344–349, 2021. arXiv preprint arXiv:2104.09968.

[7] F. Höppner, "Less is more: similarity of time series under linear transformations," Proceedings of the 2014 SIAM International Conference on Data Mining, pp. 560–568, 2014.

[8] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: a novel symbolic representation of time series," Data Mining and knowledge discovery, vol. 15, no. 2, pp. 107–144, 2007.

[9] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 206–215, 2004.

[10] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Deep adaptive input normalization for time series forecasting," IEEE transactions on neural networks and learning systems, vol. 31, no. 9, pp. 3760–3765, 2019.

[11] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: a survey and empirical demonstration," Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 102–111, 2002.

[12] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," Knowledge and information Systems, vol. 3, no. 3, pp. 263–286, 2001.

[13] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," Proceedings of the VLDB Endowment, vol. 1, no. 2, pp. 1542–1552, 2008.

[14] H. A. Dau et al. "The UCR time series archive," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 6, pp. 1293–1305, 2019.

[15] P. Senin and S. Malinchik, "Sax-vsm: Interpretable time series classification using sax and vector space model," 2013 IEEE 13th international conference on data mining, pp. 1175–1180, 2013.

[16] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y.Wah, "Time-series clustering–a decade review," Information systems, vol. 53, pp. 16–38, 2015.

[17] D. D. Paepe, D. N. Avendano, and S. V. Hoecke, "Implications of Z-Normalization in the Matrix Profile," International Conference on Pattern Recognition Applications and Methods, pp. 95–118, Springer, Cham, 2019.

[18] C. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi, "A novel bit level time series representation with implication of similarity search and clustering," In Pacific-Asia conference on knowledge discovery and data mining, pp. 771–777, Springer, Berlin, Heidelberg, 2005.

[19] A. Bagnall, et al., "A bit level representation for time series data mining with shape based similarity," Data mining and knowledge discovery, vol. 13, no. 1, pp. 11–40, 2006.

[20] P. Senin, "Z-normalization of time series," https://jmotif.github.io/sax-vsm_site/morea/algorithm/znorm.html [accessed 12-April-2023]

[21] Normalization, https://www.codecademy.com/article/normalization [accessed 12-April-2023]

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

[23] K. Cho, et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[24] M. Santini, "Advantages Disadvantages of k-Means and Hierarchical clustering (Unsupervised Learning)," http://santini.se/teaching/ml/2016/Lect_10/10c_UnsupervisedMethods.pdf [accessed 12-April-2023]

[25] A. Bagnall, et al., "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," Data mining and knowledge discovery, vol. 31, no. 3, pp. 606–660, 2017.

[26] H. Ismail Fawaz, et al., "Deep learning for time series classification: a review," Data mining and knowledge discovery, vol. 33, no. 4, pp. 917–963, 2019.

[27] Root-mean-square deviation, https://en.wikipedia.org/w/index.php?title=Root-mean-square_deviationoldid=1108025020 [accessed 12-April-2023]

[28] X. Wang, et al., "Experimental comparison of representation methods and distance measures for time series data," Data Mining and Knowledge Discovery, vol. 26, no. 2, pp. 275–309, 2013.

[29] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 6, no. 2, pp. 107–116, 1998.

[30] Joe H Ward Jr, "Hierarchical grouping to optimize an objective function," Journal of the American statistical association, vol. 58, no. 301, pp. 236–244, 1963.

[31] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," Journal of the royal statistical society. series c (applied statistics), vol. 28, no. 1, pp. 100–108, 1979.

[32] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," Acm Sigmod Record, vol. 23, no. 2, pp. 419–429, 1994.

[33] "City of Melbourne - Pedestrian Foot Traffic." www.pedestrian.melbourne.vic.gov.au [accessed 12-April-2023].

[34] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, "Automatic anomaly detection in the cloud via statistical learning," arXiv preprint arXiv:1704.07706, 2017.

[35] N. Zou, J. Wang, G.L. Chang, J. Paracha, "Application of advanced traffic information systems: field test of a travel-time prediction system with widely spaced detectors," Transportation Research Record, vol. 2129, no. 1, pp. 62–72, 2009.

[36] T. J. Lee, J. Gottschlich, N. Tatbul, E. Metcalf, and S. Zdonik, "Greenhouse: A Zero-Positive Machine Learning System for Time-Series Anomaly Detection," arXiv preprint arXiv:1801.03168, 2018.

[37] Deeplearning4j, https://deeplearning4j.konduit.ai/, [accessed 12-April-2023].