Article

Anomaly Detection Paradigm for Multivariate Time Series Data Mining for Healthcare

Abdul Razaque [1,*] **, Marzhan Abenova** [1] **, Munif Alotaibi** [2,*] **, Bandar Alotaibi** [3,4,*] **, Hamoud Alshammari** [5] **, Salim Hariri** [6] **and Aziz Alotaibi** [7]

1   Department of Cyber Security, International Information Technology University, Almaty 050000, Kazakhstan
2   Dahaa Research Group, Department of Computer Science, Shaqra University, Shaqra 11961, Saudi Arabia
3   Sensor Networks and Cellular Systems (SNCS) Research Center, University of Tabuk, Tabuk 47731, Saudi Arabia
4   Department of Information Technology, University of Tabuk, Tabuk 47731, Saudi Arabia
5   Computer and Information Science College, Jouf University, Sakakah 72388, Saudi Arabia
6   Department of Electical and Computer Engineering, University of Arizona, Tucson, AZ 85721, USA
7   Computers and Information Technology College, Taif University, Taif 21974, Saudi Arabia
*   Correspondence: a.razaque@iitu.edu.kz (A.R.); munif@su.edu.sa (M.A.); b-alotaibi@ut.edu.sa (B.A.)

Abstract: Time series data are significant, and are derived from temporal data, which involve real numbers representing values collected regularly over time. Time series have a great impact on many types of data. However, time series have anomalies. We introduce an anomaly detection paradigm called novel matrix profile (NMP) to solve the all-pairs similarity search problem for time series data in the healthcare. The proposed paradigm inherits the features from two state-of-the-art algorithms: Scalable Time series Anytime Matrix Profile (STAMP) and Scalable Time-series Ordered-search Matrix Profile (STOMP). The proposed NMP caches the output in an easy-to-access fashion for single- and multidimensional data. The proposed NMP can be used on large multivariate data sets and generates approximate solutions of high quality in a reasonable time. It is implemented on a Python platform. To determine its effectiveness, it is compared with the state-of-the-art matrix profile algorithms, i.e., STAMP and STOMP. The results confirm that the proposed NMP provides higher accuracy than the compared algorithms.

Keywords: time series; NMP; anomalies; data mining; similarities in time series; clustering

1. Introduction

Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner [1]. The relationships and summaries derived from a data mining exercise are often referred to as models or patterns (e.g., linear equations, laws, clusters, tables, tree structures, and repeated patterns of time series [2,3]). Data mining professionals tend to use fast, unsupervised methods in the early stages of the data mining process. Thus, data mining tasks (e.g., motif discovery, discord discovery, clustering, and segmentation) should be handled for time series data [4–8]. However, time series have anomalies due to similarities [9,10]. The scalable and sturdy similarity join algorithm has been proposed for handling time series anomalies [11]. However, it has a number of issues in the text domain, including community discovery, duplicate detection, collaborative filtering, clustering, and refinement of queries. Almost all of the similarity-handling algorithms for time series data either solve these problems partially or fail to address the issues [12]. STAMP is scalable for time series all-pairs-similarity-search. It achieves high-quality solutions in a realistic time [13]. However, the computational complexity of the algorithm is still higher. Additionally, this method proposed probabilistic K-nearest neighbor (PKNN) and hierarchical clustering workloads. However, the method used the expensive measure of

dynamic time warping (DTW) computation for filtering. Time series data mining prediction is depicted in Figure 1.
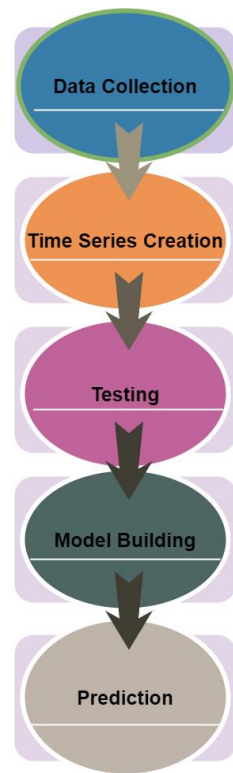

Figure 1. Time series data mining prediction.

A principal component analysis-based algorithm has been introduced for similarity search on multivariate time series (MTS) data sets [14]. However, it does not reflect changes in the relationships between the variables. There has been relatively little progress in handling anomalies of similarities in time series data. Handling these anomalies requires a robust algorithm. Therefore, the proposed novel NMP algorithm incorporates the best features of known STAMP and STOMP algorithms. It has the capability of precomputing and storing some information that can be used for data mining tasks, and the data mining process can thus be greatly accelerated.

The NMP calculates and stores effective and easy-to-access all-pair-similarity-search information, and this information can be used in a variety of data mining activities that range from well-defined tasks (e.g., motif discovery) to more open-ended tasks (e.g., representation learning). The datasets for multivariate time series with anomalies have been used and downloaded from the Kaggle [15] for proving the validity of the proposed NMP.

1.1. Research Problem Exploration

All-pairs-similarity search is a critical problem when the data objects have nearest neighbors. Many algorithms have been proposed for identification in the text domain, collaborative filtering, clustering, community discovery, duplicate detection, and query refinement. However, all of the existing text processing algorithms have time series data mining similarity problems. There has been relatively little development on all-pairs-similarity-search time series subsequences. To handle this problem, state-of-the-art solutions have been proposed, which handle the following tasks in time series data prediction:

- Providing the full join method, which eliminates the need to specify a similarity threshold.
- Recovering the top K-nearest neighbors or the nearest neighbor for each object if that neighbor is within a user-supplied threshold.

However, these solutions provide partial support for time series data in data mining. Thus, there is a need for a scalable algorithm to handle data mining tasks in time series to address large data sets and generate approximate solutions of higher quality in a reasonable amount of time.

1.2. Research Significance

The lack of consistency and interpretations in a business may cause the loss of business. This situation occurs when prediction is not constant due to anomalies available in time series data. To address this issue, the NMP algorithm is introduced, which is compatible with large data sets and produces approximate solutions of higher quality in a short time. The significance of this work is to address several data mining tasks, such as motif discovery, discord discovery, long-term evolution (LTE) discovery, semantic segmentation, and clustering. By efficiently handling these tasks, business-related search activities can be greatly improved.

1.3. Research Contribution

The contributions of this article are summarized as follows:

- The NMP uses an ultrafast similarity search process based on the z-normalized Euclidean distance as a subroutine, exploiting the redundancies between overlapping subsequences to achieve dramatic speedup and low space overhead.
- The proposed NMP provides no false positives or false negatives. This property is important in many domains.
- The NMP provides lower space complexity, namely, $O(n)$, which helps in handling several data mining tasks (e.g., motif discovery, discord discovery, LTE discovery, semantic segmentation, and clustering).

1.4. Proposed Solution

A novel matrix profile algorithm is proposed to solve the all-pairs-similarity search problem for time series data. The proposed algorithm leverages the significant features from the STAMP [13] and STOMP [16] algorithms to provide a lower time for time series data search. The proposed NMP algorithm involves different similarity search measures (shape-based [17], edit-based, feature-based, and structure-based). The proposed method also consists of two algorithms, the distance determination process for neighboring nodes and immediate neighbor detection, which help to determine the similarity of the time series data.

1.5. Research Structure

The remaining parts of this paper are structured as follows: Section 2 includes the salient features of the related work. Section 3 describes the matrix profile (MP) plan of the research. Section 4 covers the implementation and results. Section 5 contains a discussion of the implementation, including its advantages and shortcomings. Finally, Section 6 concludes the paper and provides future analysis.

2. Background and Related Literature

In this section, the salient features of existing methods are extensively discussed. There is a lot of literature available for identifying time-series anomalies, which is classified into two categories. The first set of strategies examines each individual time series using univariate models, whereas the second uses several time series as a single entity. Furthermore, current anomaly detection systems may be classified into two paradigms: reconstruction-based and forecasting-based models. This section summarizes major studies on time-series anomaly detection and delves further into these two paradigms.

2.1. Univariate Anomaly Detection

To model normal/anomaly event patterns, traditional techniques such as wavelet analysis, hypothesis testing, Singular Value Decomposition (SVD), and Autoregressive Integrated Moving Average (ARIMA) often use handmade features Zhao et al. [18]. Recently, Netflix published a comprehensive principal component analysis-based scalable anomaly detection system that has been effective in various real-world circumstances. Additionally, Twitter has made a seasonality-based anomaly identification technique available that employs the seasonal hybrid extreme study deviation test (S-H-ESD). Time-series anomaly detection has a solid foundation thanks to recent developments in neural networks. The spectral residual convolutional neural network (SR-CNN) Liu et al. [19] combines the advantages of Spectral Residual (SR) and convolutional neural network to provide state-of-the-art performance on univariate time series anomaly identification. DONUT is an unsupervised anomaly detection approach based on Variational Auto-Encoder (VAE).

2.2. Multivariate Anomaly Detection

Rebuilding-based models: By reproducing the original input based on specific latent factors, it learns the illustration for the whole time series. It obtains the features from the unsupervised model by mapping the characteristics of an instance to integrated visible neurons, which are then employed by an autoencoder to reconstruct the features. GANs (Generative Adversarial Networks) are also commonly utilized in multivariate time series anomaly identification. A novel GAN-based anomaly detection approach is utilized in multivariate time-series anomaly detection that uses the GAN-trained discriminator as well as residuals between generator-reconstructed data and real samples. Unpredictable instances might cause deterministic techniques to fail; to solve this, stochastic models for multivariate time-series anomaly detection are utilized. To capture the common patterns behind data, it learns robust representations of multivariate time series with stochastic variable connection and planar normalizing flow. A pattern with a low modernization probability is considered an anomaly under their model. Zhu et al. [16] introduced the STOMP algorithm that improved the scalability of the motif discovery. The proposed algorithm produced illegal perceptions in seismology and other fields. Michael and Karsin [17] proposed several techniques for optimizing self-joining using a GPU, which included a GPU-efficient index that employs a bounded search, a batching scheme to accommodate large result sets, and duplicate search removal with low overhead. In addition, they proposed a performance model that reveals the bottlenecks related to the result of a given data set size and enables the selection of a batch size that mitigates two sources of performance degradation. However, the proposed model fails to calculate the distance between points, and it degrades the performance of index searches when the search distance is increased. Wan and Davis [20] proposed an auto-distance covariance function for serial dependence evaluation to support the estimate residuals. The proposed method is based on time series model classification. The proposed method correctly identified misconfigurations when it was applied superfluously. However, the proposed model failed to justify the claimed idea. Furthermore, this process requires higher complexity. Kalmykov and Kalmykov [21] exploited a similarity threshold more aggressively to limit the set of candidate pairs that are considered to reduce the amount of information indexed initially. The proposed method is based on the discrete Fourier transform (DFT). However, the method is complex, with many parameters to adjust.

Prediction-based models: Anomalies are detected based on prediction errors. These models use non-parametric thresholding and unsupervised features to evaluate predictions provided by the Long Short-Term Memory (LSTM) network. They develop an automated anomaly detection system to monitor the spacecraft's telemetry data. For multivariate time series, a real-time anomaly detection system based on Hierarchical Temporal Memory and Bayesian Network can also be utilized. Through an end-to-end learning framework, non-temporal dimensional reduction and recurrent auto-encoders may be used to handle the challenge of multivariate time series. Tsuchiyama and Junichi [22] proposed an efficient

method to detect large-scale multimedia data using waveform similarity that overcomes the disadvantages of existing detection methods. The proposed mechanism consists of an efficient supervised multimodal hashing method that focuses on directly employing semantic labels to monitor the hashing learning process. Rubinstein [23] proposed the Partition-based Similarity Search (PSS) algorithm, which uses a static partitioning algorithm that places dissimilar vectors into different groups and balances the comparison workload with a circular assignment. Wang et al. [24] proposed the label consistent matrix factorization hashing (LCMFH) algorithm, which maps heterogeneous information that focuses on an inert space and then adjusts the idle space to an inactive semantic space obtained from class labels. However, LCMFH does not consider the shortcomings of quantization, which corrupts the separation of hash codes. Johnson et al. [25] proposed an algorithmic structure of similarity search methods that achieves near-optimal performance (NOP) on a graphical process that helps to perform close ideal execution on graphical processing units. However, the performance of proposed approach is greatly affected due to the complex nature of the algorithm. All of the existing proposed methods for time series are either complex or do not scale well for similarity search from a data mining perspective. Our proposed algorithm has less complexity and is highly compatible with similarity search from a data mining perspective.

3. Proposed Novel Matrix Profile for Similarity Search

The proposed NMP consists of three phases:

- Inputting the Time Series (ITS);
- All-pairs-Similarity Search Process (ASSP);
- Distance Determination and Matrix Profile Index (DD&MPI).

3.1. Inputting the Time Series (Its)

It performs the collection of observations sequentially over time and occurs in a variety of fields (e.g., economics, financial, physical, marketing, demographic, engineering). These series can be univariate or multivariate, which is when several series simultaneously span multiple dimensions within the same time range. Figure 2 depicts the proposed model for time series data. It involves components that are common to most time series mining tasks.
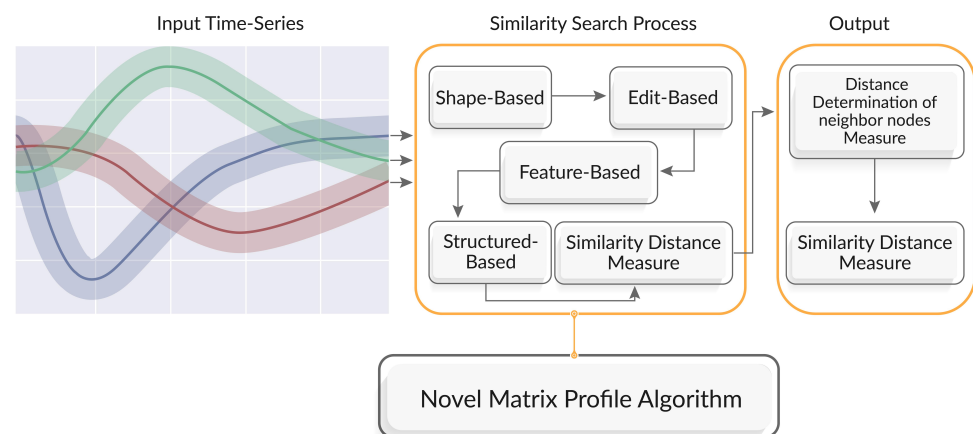
Figure 2. Proposed model for time series data set supported with NMP algorithm.

3.1.1. Data Preprocessing

Issues of noise filtering, outlier handling, scale concerns, normalization, and resampling of time series usually arise in time series in realistic situations. All of these problems are usually handled by preprocessing the data. Ma [26] proposed a fast algorithm for filtering noise for time series prediction using Recurrent Neural Networks (RNNs). Cai et al. [27] exploited the Sliding Window Prediction (SWP) model to perform the correction of outliers.

Additionally, they used standard temperature time series data gathered from the Nanjing University of Information Science and Technology (NUIST) weather stations.

The next concern relates to different variations in the scale of time series. As proposed by Pelletier et al. [28], a linear transformation of the amplitudes can handle this problem. Smyl [29] proposed a dynamic computational graph neural network (DCGNN) system, which received an award in the M4 competition. For the preprocessing, the time series data were normalized and divided into different ranges to use as input for obtaining the desired output.

Because data normalization is a fundamental preprocessing step, we will find an appropriate method to address time series normalization tasks. Here is one of the traditional data normalization methods:

Definition 1. *A time series T (please refer to Appendix A for more information) is called Z-normalized when its mean μ is 0 and its standard deviation σ is 1. The normalized version of $T = t_1, ..., t_{|T|}$ is computed as follows:*

$$c^n = \left\{ \frac{t_1 - \mu}{\sigma}, ..., \frac{t(Tl) - \mu}{\sigma} \right\} \tag{1}$$

where T^n is the z-based normalized time series of T; $t_1, \cdots, t_{|T|}$ is a sequence of numbers $t_i \in \mathbb{R}$, where $i \in N$ represents the position in T; Tl is the length or size of time series T; and σ is the standard deviation of time series T.

This approach works well in stationary settings where the exact minimum and maximum values of the time series are unknown, but it struggles with nonstationary time series where the mean and standard deviation change over time.

On the other hand, this method is useful when the minimum and maximum values of an attribute are undefined and can be extended to stationary time series, in other words, time series in which statistical properties such as the mean, variance, and autocorrelation remain constant over time.

Moreover, z-normalization is an essential operation in several applications because it allows similarity search irrespective of shifting and scaling.

3.1.2. Data Representation

Time-series data representation emphasizes the following essential characteristics:

- Temporal representation: storing a temporal point on the displayed data.
- Spectral representation: designing the data in the frequency domain.
- Other representations: implementing different modifications not related to the above.

This approach is described in a concise way and gains additional advantages, such as efficient storage, speedup of processing, and implicit noise removal. The essential characteristics of data representation require the following for any representation:

- A significant drop in the data dimensionality.
- An emphasis on the essential shape features for both global and local scales.
- Lower computational costs for the computational representation.
- Better restoration quality for the reduced representation.
- Implicit noise handling or insensitivity to noise.

By studying several properties of a sequence at the same time, issues such as amplitude, scaling, temporal warping, noise, and outliers can be prevented. The following transformations are considered for a given time series T with n data points $T = \{t_1, ..., t_n\}$: *Amplitude shifting*: The series $G = \{g_1, ..., g_n\}$ obtained by a linear amplitude shift of the original series is given by:

$$g_i = t_i + k \tag{2}$$

where g_i is the series of T; t_i is a time series such that $t_i \in T$; and k is a constant, where $k \in \mathbb{R}$.

Uniform amplification: The series G obtained by multiplying the amplitude of the original series is specified as:

$$g_i = k \times t_i \tag{3}$$

Uniform time scaling: The series $G = \{g_1, ..., g_m\}$ produced by a uniform change in the time scale of the original series is given as follows:

$$g_i = t_{|k \times i|} \tag{4}$$

Dynamic amplification: The series G obtained by multiplying the original series by a dynamic amplification function that is assigned is:

$$g_i = h(i) \times t_i \tag{5}$$

where $h(i)$ is a function such that $\forall t \in [1, 2, ..., e], h'(t) = 0$ if and only if $t_i' = 0$.

Dynamic time scaling: The series G obtained by a dynamic change in the time scale is generated by:

$$g_i = t_{h(i)} \tag{6}$$

where $h(i)$ is a positive, strictly increasing function such that h is:

$$\mathbb{N} \longrightarrow [1, ..., e] \tag{7}$$

where e is the length of $\mathbb{N}$.

Additive Noise: The series G obtained by adding a noisy component to the original series is given by:

$$g_i = t_i + \varepsilon_i \tag{8}$$

where $t_i + \varepsilon_i$ and ε_i are independent identically distributed white noise.

Outliers: The series G is obtained by adding outliers at random positions. Formally, these are a given set of random time positions expressed by:

$$\rho = \left\{ k \in [1, ..., e] \right\} \tag{9}$$

where ρ is a random time position and $g_k = \varepsilon_k$, where ε_k is independent identically distributed white noise.

Definition 2. *The similarity measure $D(T, U)$ between time series T and U is a function that takes two time series as inputs and returns the distance d between the series. This distance has to be nonnegative; i.e., $D(T, U) \geq 0$.*

If this measure satisfies the *additional symmetry property $D(T, U) = D(U, T)$* and *subadditivity $D(T, V) \leq D(T, U) + D(U, V)$* (also known as the *triangle inequality*), the distance is said to be a *metric*.

The similarity measure $D(T, G)$ should be robust to any combination of these transformations. This property leads to our formalization of four general types of robustness. We introduce properties that express robustness for scaling (amplitude modifications), warping (temporal modifications), noise, and outliers. Let S be a collection of time series, and let H be the maximal group of homeomorphisms under which S is closed.

A similarity measure D on S is called scale robust if it satisfies the following:

Property 1. *For each $T \in S$ and $\alpha > 0$, there is a $\delta > 0$ such that $||t_i - h(t_i)|| < \delta$ for all $t_i \in T$, which implies $D(T, h(T)) < \alpha$ for all $h \in H$.*

We call a similarity measure *warp robust* if the following holds:

Property 2. *For each $T = \{t_i\} \in S, T' = \{t_{h(i)}\}$ and $\alpha > 0$, there is a $\delta > 0$ such that $||i - h(i)|| < \delta$ for all $t_i \in T$ implies that $D(T, T') < \alpha$ for all $h \in H$.*

We call a similarity measure noise robust if it satisfies the following property:

Property 3. *For each $T \in S$ and $\alpha > 0$, there is a $\delta > 0$ such that U with $p(\varepsilon) = N(0, \delta)$ implies $(T, U) < \alpha$ for all $U \in S$.*

$$U = T + \varepsilon \tag{10}$$

where U is a function such that $U \in T$ and ε is independent identically distributed white noise.

We call a measure *outlier robust* if the following holds:

Property 4. *For each $T \in S, K = \{rand[1, ..., e]\}$ and $\alpha > 0$, there is a $\delta > 0$ such that if $|K| < \delta$, $U_{(k \in K)} = \varepsilon_k$ and $U_{(k \notin K)}$, then $D(T, U) < \alpha$ for all $U \in S$.*

$$U_{k \notin K} = T_k \tag{11}$$

where $U_{k \notin K}$ implies $U_{k \in K} = \varepsilon_k$.

3.2. All-Pairs Similarity Search Process

It defines how any pair of time series is distinguished or matched and how an intuitive distance between two series is formalized. This measure should establish a notion of similarity based on perceptual criteria, thus allowing the recognition of perceptually similar objects even when they are not mathematically identical. Additionally, it searches for similarities in data objects. Every time series mining task requires a subtle notion of similarity between time series that is based on the more intuitive notion of shape.

It involves different types of similarity search measures:

- Shape-based;
- Edit-based;
- Feature-based;
- Structure-based.

3.2.1. Shape-Based Similarity

Shape-based distances compare the overall shape of the series. On the other hand, dynamic time warping (DTW) provides a grasp of the local biases of the time axis by using nonuniform time warping, as described by Hong et al. [30]. This measure is capable of matching different parts of a time series by allowing the time axis to be warped. The shortest warping path in a distance matrix determines the optimum alignment. A warping path is a set of contiguous matrix indices that describe the mapping of two time series. The optimal path minimizes the global warping expense, even though there is an exponential number of potential warping paths. DTW can be determined using dynamic time-complexity programming $O(n2)$. The notion of the upper and lower envelopes is introduced in the approach proposed by Rubinstein and Zhao [31], which represents the maximum allowed warping technique, where the time complexity becomes $O(n)$. A temporal restriction can also be imposed on the duration of the DTW window. These strategies have been shown to increase not only the speed but also the level of accuracy, as they prevent extended warping from pathological matching.

These measures do not match any of the types of robustness. Even if the problems of scaling and noise can be handled in a preprocessing step, the warping and outlier issues must be addressed with more sophisticated techniques. The use of elastic measures can provide an elegant solution to both problems.

3.2.2. Edit-Based Similarity

The edit-based method is used to characterize the distance between two strings. The underlying idea is that the minimum number of operations needed to transform one string into another, with insertion, deletion, and substitution, can represent the distance between strings.

The longest common subsequence (LCSS) algorithm is used to handle outliers or noisiness in matching two time series, as discussed in Vishwakarma et al. [32]. For point matching and a warping threshold δ, the LCSS distance utilizes the threshold parameter ε. To compare the shapes of subsequences between two series, the NMP utilizes the z-normalized Euclidean distance.

Property 5. *The z-normalized Euclidean distance between two sequences of length m is, in effect, a function of the correlation between the two sequences as given below:*

$$d_{x,y} = \sqrt{2m\big(1 - corr(x,y)\big)} \tag{12}$$

where d is the z-normalized Euclidean distance between two sequences x and y, m is the length of the two sequences x, y and corr() is the Pearson correlation coefficient between the two sequences x, y.

Proof. First, we illustrate the following properties of the inner product of the z-normalized sequence:

$$\sigma_x^2 \underset{\frac{\sum_i^m (x_i - \mu x)^2}{\sigma x}}{} \tag{13}$$

where σ_x^2 is the variance of the sequence x.

$$d(x,y)^2 = \sum_i^m \left(\frac{x_i - \mu x}{\sigma x}\right)^2 \tag{14}$$

Using Equation (14), a function of the correlation between the two sequences can be proved as:

$$
\begin{aligned}
d(x,y)^2 &= \sum_i^m \left(\frac{x_i - \mu x}{\sigma x} - \frac{y_i - \mu y}{\sigma y}\right)^2 \\
d(x,y)^2 &= \sum_i^m \left(\frac{x - \mu x}{\sigma x}\right)^2 - 2\sum_i^m \left(\frac{x - \mu x}{\sigma x}\right)\sum_i^m \left(\frac{y - \mu y}{\sigma y}\right) + \sum_i^m \left(\frac{y - \mu y}{\sigma y}\right)^2 \\
d(x,y)^2 &= 2m\left(1 - \frac{1}{m}\sum_i^m \left(\frac{x - \mu x}{\sigma x}\right)\right)2m\left(1 - \frac{1}{m}\sum_i^m \left(\frac{x - \mu x}{\sigma x}\right)\right) \\
d(x,y)^2 &= 2m\big(1 - corr(x,y)\big) \\
d(x,y) &= \sqrt{2m\big(1 - corr(x,y)\big)}
\end{aligned}
\tag{15}
$$

The correlation property between the two sequences given in Equation (13) is proved. Thus, Equation (15) shows equality.

The Pearson correlation coefficient (PCC) calculates the strength of the linear association between two variables. Let x and y be random variables. The PCC is defined as:

$$
\begin{aligned}
corr{x,y} &= \frac{\left(E(x) - \mu x\right)\left(E(y) - \mu y\right)}{\sigma_{xy}} \\
&= \frac{\sum_{i=1}^m Q_{x,y} - m\mu_x\mu_y}{m\sigma_x\sigma_y}
\end{aligned}
\tag{16}
$$

where E is the expected distance between the two sequences x and y and $Q_{x,y}$ is the dot product of time series $T_{x,m}$ and $T_{y,m}$.

The proposed connection of the z-normalized Euclidean distance with the Pearson correlation coefficient is expressed by the following equation:

$$
\begin{aligned}
d_{x,y} &= \sqrt{2m\left(1 - corr(x,y)\right)} \\
&= \sqrt{2m\left(1 - \frac{Q_{x,y} - m\mu_x\mu_y}{m\sigma_x\sigma_y}\right)}
\end{aligned}
\tag{17}
$$

□

Definition 3. *The z-normalized Euclidean distance $d_{x,y}$ of two time series subsequences $T_{x,m}$ and $T_{y,m}$ can be evaluated as follows:*

$$
d_{x,y} = \sqrt{2m\left(1 - \frac{Q_x, y - m\mu_x\mu_y}{m\sigma_x\sigma_y}\right)}
\tag{18}
$$

where μ_x is the mean of $T_{x,m}$, μ_y is the mean of $T_{y,m}$, σ_x is the standard deviation of $T_{x,m}$ and σ_y is the standard deviation of $T_{y,m}$.

Considering that the correlation is limited to the range $[-1, 1]$, $d_{x,y}$ can fall within the range $\left[0, 2\sqrt{m}\right]$ between two sequences of length m, where zero implies an ideal match and $2\sqrt{m}$ corresponds to the worst possible match. As a consequence, the upper bound of $2\sqrt{m}$ can be used to normalize distances to the range $[0, 1]$, which can help us to equate matches of different lengths and enable us to identify and repeat thresholds to define degrees of resemblance by using $d_{x,y}$. Rather than defining a threshold that is dependent on m, we can define a more uniform similarity threshold for sequences of any length in this way. Linardi et al. [33] found that the normalization factor $\sqrt{m}$ could compare matches of different lengths, but no reference to the underlying mathematics was made. The distance bounds for Z_{ed} of 0 and $2\sqrt{m}$ lead to 1 and -1 correlation coefficients, respectively. This approach assumes that $d_{x,y} = 0$ and $d_{x,z} = 2\sqrt{m}$ for any sequence x of length m with $\sigma_x \neq 0$. In the case of:

$$
y = ax + b
$$
$$
z = -ax + b
$$

where z is the sequence of time series, a, b are any values, for $a > 0$.

Let $s \in R^m$ be a sequence, $n \in R^m$ and $n\prime \in R^m$ be the two noise sequences that are sampled using normal distribution $N(0, \sigma_N^2)$. Then, the estimated distance between the two sequences obtained by applying the noise to the base sequence can be represented as follows:

$$
x = y = s + n + n\prime
$$

where s is an unknown constant for x and y and n is a normal distribution, where $n \sim N(0, \sigma_N^2)$.

$$
\mathbb{E}\left[d_{x,y}^2\right] = (2m + 2)\frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2}
\tag{19}
$$

where σ_N^2 is the variance of the noise and $\sigma_S^2 + \sigma_N^2$ is the predicted variance of the noisy sequence.

We now determine the estimated effect of the noise. Henceforth, we assume sequences to be non-random variables and illustrate this idea by denoting them as x and y. Furthermore, Equation (20) is also updated as:

$$\mathbb{E}\left[d_{x,y}^2\right] = \mathbb{E}\left[(x_1 - y_1)^2 + ... + (x_m - y_m)^2\right]$$
$$= m \cdot \mathbb{E}\left[(x - y)^2\right]$$
$$\mathbb{E}\left[d_{x,y}^2\right] = m \cdot \mathbb{E}\left[\left(\frac{x - \mu x}{\sigma_x} - \frac{y - \mu y}{\sigma_y}\right)^2\right]$$

(20)

Since x and y are the products of identical variables, the two variables have equal variance.

$$\sigma_x^2 = \sigma_y^2 = \sigma_S^2 = \sigma_N^2$$

(21)

Next, we decompose the additives μ_x and μ_y from the initial sequences μ_s and the noise effect. Here, we use n as a random variable from the noise distribution. We observe that μ_s can be used as a constant by connection with the mean of the base sequence.

$$\mu_x = \mu_y = \mu_s + \frac{n_1 + ... + n_m}{m}$$
$$= \mu_s + \mu_n$$
$$\mu_N \sim N\left(0, \frac{\sigma_N^2}{m}\right)$$

(22)

We perform the same decomposition for x and y, where s is an undefined constant from the base sequence:

$$x = y = s + n$$

(23)

$$n \sim N(0, \sigma_N^2)$$

The constant expressions are cancelled, and the distributions are merged, yielding the following:

$$\mathbb{E}\left[d_{x,y}^2\right] = m \cdot \mathbb{E}\left[\left(\frac{n_x - n_y - \mu N_x + \mu N_y}{\sqrt{\sigma_S^2 + \sigma_N^2}}\right)^2\right]$$
$$\mathbb{E}\left[d_{x,y}^2\right] = m \cdot \mathbb{E}[(V^2)]$$
$$V \sim N\left(0 \cdot \frac{2 + 2m}{m} \cdot \frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2}\right)$$

(24)

Substitute the value of V in Equation (24):

$$\mathbb{E}\left[d_{x,y}^2\right] = m \cdot \mathbb{E}\left[N\left(0, \frac{2 + 2m}{m} \cdot \frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2}\right)\right]^2$$
$$\mathbb{E}\left[d_{x,y}^2\right] = m \cdot \mathbb{E}\left[N\left((2m + 2) \cdot \frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2}\right)\right]^2$$

(25)

Theorem 1. *Given a time series, the position that gives the maximum average point-to-point distance is the position that has the highest absolute Z-normalized value.*

Proof. After Z-normalization, the mean and standard derivation of a time series are 0 and 1, respectively. For any fixed point a, the expected distance between this factor and other factors is:

$$E[(X - a)^2] = E[X^2 - 2Xa + a^2]$$
$$E[(X - a)^2] = E[X^2] - 2aE[X] + a^2E[1]$$
$$E[(X - a)^2] = E[(\mu)^2] - 2a\mu + a^2$$
$$E[(X - a)^2] = var(X) - 2a\mu + a^2 = 1 + a^2$$

(26)

where E is the expected point-to-point distance, a is any fixed point, and X is the expected maximum value. Consequently, the average point-to-point distance from a fixed point a to different points is $1 + a^2$. Therefore, the point with the highest absolute Z-normalized value can be the predicted point-to-point distance. □

Lemma 1. *The optimal ordering for early abandonment of the Euclidean distance is ordering by the absolute Z-normalized value.*

Proof. An outline of all point-to-point distances is the Euclidean distance, and it is a monotonically nonreducing function. From Theorem 1, the predicted contribution to the summation can be increasingly expanded to large amounts as viable if we order the points in the series by their absolute Z-normalized values for each dimension. The sum of points used for early abandonment is then minimized, and the result determines the optimum order. In line with Lemma 1, we can obtain a perfect order in early abandonment by ordering the absolute Z-normalized values. Next, we can empirically enumerate the number of point-to-point predictions in the early abandonment process. □

3.2.3. Feature-Based Similarity

Feature-based distances extract features that describe aspects of the series that are then compared with any type of distance function. This measure is mostly used to determine the local similarities between patterns. However, when handling longer time series, it might be profitable to find similarities on a global scale. This approach has the following properties:

- It uses an ultrafast similarity search process and the z-normalized Euclidean distance as a subroutine.
- It exploits the redundancies between overlapping subsequences.
- It provides lower space complexity to handle several data mining tasks.

3.2.4. Structure-Based Similarity

The main purpose of structure-based similarity is to find higher-level structures in the series. This approach compares structures on a global scale. It is divided into two further subcategories:

- Model-based distances;
- Compression-based distances.

1. *Model-based distances*

   This approach applies a model to different series and then compares the parameters of the underlying model. Similarity can be calculated by modeling the underlying time series. This determines the probability of one time series by using another underlying model. Any type of parametric temporal model can be used. Furthermore, the distance determination process for neighboring nodes is given in Algorithm 1.

   In Algorithm 1, the distance of the neighbor node is determined. In step 1, the variables are initialized. The input and output are given at the beginning of the algorithm. In step 2, the time series length is detected from the time series data. Steps 3–4 define the memory allocation process, and the initial matrix profile and matrix profile index are stored in memory. Steps 5–7 calculate the mean value and standard deviation. Step 8 is used to perform the vector dot product. In step 9, the distance profile index is determined from the time series. In steps 2–13, the distance of each neighbor is calculated and finally stored in an array.

Algorithm 1 Distance Determination Process

Input: $\{T_{inf}\}$ in

Output: $\{D_{ne}\}$ out

1: **Initialization:** $\{$ T_l: *Time Series Length; M: Memory; M_i: Initial Matrix Profile; M_{pi}: Matrix Profile Index; T_{inf}: Time Series Information; D_{Pi}: Distance Profile Index; M_v: Mean Values; σ: Standard Deviation; Sum_{T_l}: Sum of Time Series; D_{T_l}: Each Data Value in the Time Series; V_{dp}: Vector Dot Product; D_{ne}: Neighbor Distance$\}$*
2: **Extract** $T_l \in T_{inf}$
3: **Set** M
4: **Allocate** $M \rightarrow M_i \& M_{pi}$
5: **Calculate** $M_v = \frac{Sum_{T_l}}{T_l}$
6: **Calculate** $\mu = (D_{T_l} - M_v)^2 \& \sum_{i=1}^{N}(D_{T_l} - M_v)^2 \& \frac{1}{N}\sum_{i=1}^{N}(D_{T_l} - M_v)^2$
7: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(D_{T_l} - M_v)^2}$
8: **Set** V_{dp}
9: **Determine** D_{pi} from T_{inf}
10: **for** $D_{pi} = 0$ to $D_{pi} \leq Tinf$ **do**
11:     $D_{pi} = D_{pi} + 1$
12:     **if** $D_{pi} = T_l$ **then**
13:         **Store** $D_{ne}[i] = D_{pi}$
14:     **end if**
15: **end for**

---

2. *Compression-based distances*

    Compression-based methods determine how easily two series can be compacted together. One distance measure is based on Kolmogorov complexity, called the compression-based dissimilarity measure (CDM). It is established based on bioinformatics findings. The fundamental notion is that concatenating and compressing similar series produces higher compression ratios for different data. If this method is successful for clustering, then it can be extended for fetal heart rate tracing. This process is effective for clustering.

    Our basic approach is to find similar pairs to compute the dot product of each normalized time series over the z-normalized Euclidean distance $d_{x,y}$. In other words, the dot product $\varrho_{i,j}$ can be evaluated in $O(1)$ when $\varrho_{i-1,j-1}$ is given:

    $$\varrho_{i,j} = \varrho_{i-1,j-1} - t_{i-1}t_{j-1} + ti + u - 1t_{j+u-1} \tag{27}$$

    where $\varrho_{i,j}$ is the dot product of $T_{i,u}$ and $T_{j,u}$; $\varrho_{i-1,j-1}$ implies $Q_{i,j} \in t_{i,j}$; $i, j \in u$; and $t_{i-1}t_{j-1}$ implies $\varrho_{i,j} \in t_{i,j}$; $i, j \in u$ and $t_{i-1}t_{j-1}$ implies $t \in T$. Thus, $i, j \in u$.

    Before finding the distance profile, we search for similarities, where each normalization of each subsequence must be normalized before it is compared to the query by defining the mean value and the standard deviation. The mean of the subsequence can be calculated by holding two running sums of a long time series with a lag of exactly m values. Similarly, the sum of the subsequence squares can be determined. Here, consistency can be determined by the following:

    $$\mu = \frac{1}{S_v}\sqrt{x_i}. \tag{28}$$

    S_v are the values of string for the long time series; and x_i are the sums of the time series T.

    The standard deviation of the time series calculated by the average of the squared deviations from the mean is shown below:

$$\sigma^2 = \frac{1}{S_v} \sum x_i^2 \mu^2 \qquad (29)$$

There is another name for them, "one pass". With the help of one pass measures, we will determine the distance profile D_{Pi}.

Given a $T_{i,u}$ and a time series T query list, the distance between $T_{i,u}$ and all subsequences is determined in T. We call this a distance profile:

Definition 4. *A distance profile D_{pi} that corresponds to query $T_{i,u}$ and time series T is a vector of the Euclidean distances between a given query subsequence $T_{i,u}$ and each subsequence in time series T. Formally,*

$$D_i = [d_{i,1}, d_{i,2}, ..., d_{i,n-u+1}], \qquad (30)$$

where $d_{i,j} = (1 \geq j \geq n - u + 1)$ is the distance between $T_{i,u}$ and $T_{j,u}$.

Once we have D_i, we will dispose of the closest neighbor to $T_{i,u}$ in time series T. Note that if query $T_{i,u}$ is a subsequence of T, the i-th distance profile position D_i is zero (i.e., $d_{i,i} = 0$), and the value is near zero only to the left and right of i. In the literature, this match is considered a trivial match. We prevent such matches by ignoring the $u/4$ duration "exclusion", in practice, the following property is set:

$$\left\{ i - \frac{u}{4} \leq j \leq i + \frac{u}{4} \leq \forall r_\alpha \right\} \qquad (31)$$

where $\forall r_\alpha$ is equal to 1.

Therefore, the nearest neighbor of $T_{(i,u)}$ can be determined by evaluating u in (D_i).

3.3. Distance Determination & Matrix Profile Index (DD & MPI)

This index is the output value of the distance determination and index detection neighbor nodes. Indexing makes it possible for large-scale databases to provide an accessible organization of data for fast retrieval. Figure 3 shows the three types of time series data that are arranged according to their characteristics, but a few time series data sets are out of reach of the neighbor nodes. Thus, such data sets are confirmed outliers and anomalies. Different confidence intervals demonstrate the probability, which comes from different samples available in the dataset.
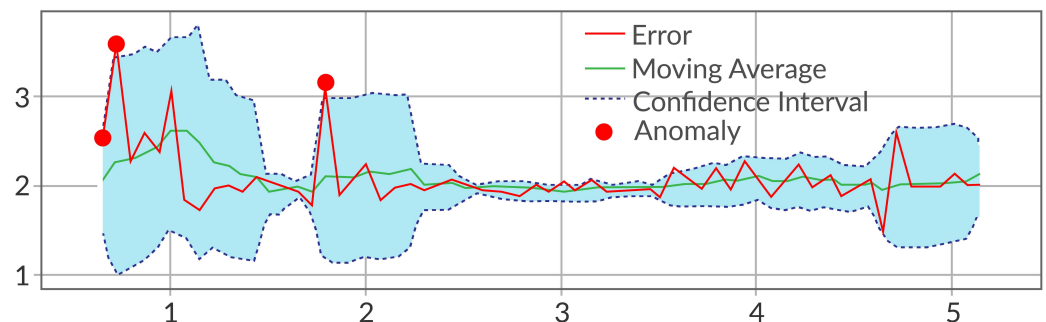

Figure 3. Determination of confirmed outliers and anomalies for time series data sets.

It is important to carefully describe the difference between time series data sets to show perceptually significant aspects of the underlying similarity. Finally, the indexing mechanism must allow ever-growing large data sets to be handled and queried effectively. The immediate neighbor detection process is further described in Algorithm 2.

Algorithm 2 Immediate Neighbor Detection Process

Input: $\{T_{inf}\}$ in
Output: $\{I_{ne}\}$ out

1: **Initialization:** $\{$ T_{inf}: *Time series, T_l: Time series length, M_s: Set the memory, M: Memory, M_i: Initial matrix profile, M_{pi}: Matrix profile index vector, M_v: Mean values of time series, S_{ts}: Subsequence from time series T_s, I_{ne}: Location of neighbor* $\}$
2: **Compute** $T_l \in T_{inf}$
3: **Initialize** $M_s, M_i \leftarrow infs, M_{pi} \leftarrow zeros$
4: **Allocate** $M \in M_i \& M_{pi}$
5: **Compute** $M_v = \frac{1}{T_l} \sum Tinf_i$
6: **Compute** $\mu == \left(T_{inf_i} - M_v\right)^2 \& \sum_{t=1}^{T_l}\left(T_{inf_i} - M_v\right)^2 \frac{1}{T_l}\sum_{t=1}^{T_l}\left(T_{inf_i} - M_v\right)^2$
7: **Compute** $\sigma = \sqrt{\frac{1}{T_l}\sum T_{inf_i}^2 - \mu^2}$
8: **Determine** M_{pi} using M_v, μ, σ from T_{inf}
9: **while** $M_{pi} \leq T_{inf}$ **do**
10:    **if** $M_{pi} = T_l$ **then**
11:       **Store** $I_{ne}[i] = M_{pi}$
12:    **end if**
13: **end while**

Algorithm 2 determines the immediate neighbor detection process. In step 1, the variables used in the algorithm are initialized. The input and output are shown at the beginning of the algorithm. In step 2, the length of the time series is calculated. In steps 3–4, we set and allocate the memory. In steps 5–7, we determine the mean values and the standard deviation of the time series. In steps 8–13, we calculate and save the location of each neighbor. The output of the algorithm is the matrix profile index.

In the basic method, we want to locate the nearest neighbor of every subsequence in T. The nearest neighbor information is stored in two meta time series, the matrix profile and the matrix profile index.

As the name suggests, the matrix profile is a profile that stores the minimum Euclidean distance of every subset of one time series with respect to another (or itself, called selfjoining). It additionally stores an associate vector called the profile index that gives the index of each nearest neighbor.

The profile index I_{ne} stores indices and therefore integers. The remaining meta time series are all real values and are stored as floating point values. Floating point stored values can be determined as:

$$V_f = (T_l - u + 1) \cdot 4 + u - 1 \tag{32}$$

where V_f are floating point stored values, n is the length of time series Tl, and m is the fixed length for each subsequence $T_{i,m}$. The next equation defines the values required for the remaining meta time series of integer values:

$$V_i = Tl - u + 1 \tag{33}$$

where V_i are stored integer values.

Dismissing the exclusion zone E_i of trivial matches, the matrix profile denotes a vector whose elements are the minima of the columns in the distance matrix. Matrix profile computation can be determined as:

$$m_{pi} = min(d_{i,j}) \tag{34}$$

where m_{pi} is the matrix profile.

The profile index I_{ne} captures the starting index j of this nearest neighbor. In the (theoretical) case of several minimizers j, the smallest minimizer will be selected:

$$I_{ne} = argmin(d_{i,j}) \tag{35}$$

where I_{ne} is the profile index. Linardi et al. [33] proposed variable-length motifs in data series that is supported using Parseval's Theorem 2.

Theorem 2. *The Euclidean distance between two signals $\vec{x}$ and $\vec{y}$ in the time series is the same as their Euclidean distance in the frequency domain.*

Proof. Let $\vec{X}$ be the discrete Fourier transform of the sequence $\vec{x}$. Then, we have:

$$\sum_{i=0,\dots,n-1} |x_t|^2 = \sum_{i=0,\dots,n \times 1} |X_f|^2 \tag{36}$$

where X_f is a linear map $f : X \to R_k$ and x_t is a linear map $f : x \to R_{k+i}$.

The discrete Fourier transform inherits the properties below from the continuous Fourier transform. Let $\Leftrightarrow$ indicate Fourier pairs, i.e.,

$$|x_t| \Leftrightarrow |X_f|, \tag{37}$$

where $|X_f|$ is the discrete Fourier transform of $|x_t|$. The discrete Fourier transform is a *linear transformation*: if the condition satisfies as:

$$|x_t| \Leftrightarrow |X_f|; |y_t| \Leftrightarrow |Y_f|$$

holds, then we obtain the following:

$$[x_t + y_t] \Leftrightarrow [X_f + Y_f]. \tag{38}$$

We then obtain:

$$[ax_t] \Leftrightarrow [aX_f]. \tag{39}$$

In addition, a shift in the time domain changes only the phase of the Fourier coefficients but not the amplitude.

$$[x_{t-t_0}] \Leftrightarrow \left[X_f exp\left(2\pi f \frac{t_0 j}{n} \right) \right] \tag{40}$$

By the above equation, Parseval's theorem gives:

$$||\vec{x} - \vec{y}||^2 = ||\vec{X} - \vec{Y}||^2. \tag{41}$$

One of the aims of the next computation is to determine how much knowledge is lost because of the dimensionality reduction and time series reconstruction periods. We aim to determine how different the restored time series is from the original by using these equations.

The Euclidean distance between the predicted and real time series determines the reconstruction error of time series T:

$$RecErr(T) = \sqrt{\sum_{t=1}^{n} (T_t - T't)^2} = \sqrt{\sum_{t=1}^{n} \left(T_t - (\beta \times t + \gamma) \right)^2} \tag{42}$$

where T'_t is the line segment $T'_t = \beta \times t + \gamma$ given for the well-approximated time series T and where β and γ are two coefficients that are a function of the linear curve.

These two parameters β and γ satisfy the two criteria below:

$$\frac{\partial RecErr(T)}{\partial \beta} = 0 \tag{43}$$

$$\frac{\partial RecErr(T)}{\partial \gamma} = 0 \tag{44}$$

By combining Equations (43) and (44), β and γ can be obtained:

$$\beta = \frac{12\sum_{t=1}^{n}\left(t - \frac{n+1}{2}\right)T_t}{n(n+1)(n-1)} \tag{45}$$

$$\gamma = \frac{6\sum_{t=1}^{n}\left(t - \frac{2n+1}{3}\right)T_t}{n(1-n)} \tag{46}$$

where T_t is the actual value at time stamp t in time series T, β and γ: selected to achieve the minimum reconstruction error $T'_t = \beta \cdot t + \gamma$. $\quad\square$

4. Experimental Results

In this section, we give the results of experiments regarding the experimental setup and performance metrics.

4.1. Experimental Setup

To confirm the performance of the proposed NMP algorithm, we use the Python platform to implement our algorithm. We compare our proposed NMP with the state-of-the-art algorithms, STAMP and STOMP. Table 1 shows the hardware and software used to implement the proposed algorithm.

Table 1. Development environment.

Parameters	Description
Personal computer	x64
Operating system	Windows 7
Processor	Intel Core i7-2670QM
RAM	6 GB
CPU GHz	2.20

Data sets. The findings are based on two datasets that are multivariate and multi-sample. The first dataset is unbalanced and consists of time series with anomalies [15] that involve 11 features with 509 k samples. The dataset is downloaded from the link: https://www.kaggle.com/drscarlat/time-series, (accessed on 20 November 2021). Anomalies make up about 0.09% of the data. Normal and abnormal data are separated, and anomaly-tagged events occur every nine time steps. The data set is segmented into 126 timestep segments, evenly separating the test batch's aberrant signals. Samples were randomly combined and supplemented to increase the feature dimension. The feature axis was cloned five times, randomly mixed, then concatenated across the dataset to increase the number of features from 11 to 55. The second data set depicts the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) pandemic, which was made public by Johns Hopkins University [34]. It describes three factors over the course of 10 months for 188 nations, ranging from the beginning day of the epidemic to the day it ended after four months.

Time series with anomalies: Time and Accuracy;
SARS-CoV-2: Recovered, Infected, and Deaths.

4.2. Performance Metrics

First, we should note that all of the implementations have been tested on common data sets of varying sizes. Here, there is no influence of NMP run-time performance on either data quality or the data inputs. To measure the algorithm's performance, we compared NMP with the STAMP and STOMP algorithms. Based on the testing results, the following metrics were considered:

- Time required for the similarity search;
- Accuracy;
- Detection efficiency.

4.2.1. Time Required for Similarity Search

The run-time computation of a particular algorithm when performing the all-pairs similarity search is calculated as:

$$T_{req} = \frac{T_{cal}}{n_{cal}^2} \times n_{new}^2 \tag{47}$$

where T_{req} is the time spent for the performance computation of T_{cal} on every given hardware configuration, T_{cal} is the computing time in one calibration run, n_{cal} is the length of one calibration run, and n_{new} is the new length of the time series T.

In Figure 4a, we show the time required for the similarity search. We generated the list of queries by attempting to extract a random data series. Random time series data set queries of lengths 18,000, 45,000, 90,000, and 135,000 were used. In Figure 4a, we observe that STOMP and STAMP show different times for time-series query detection, which is 115 for STOMP and 118 for STAMP milliseconds. Thus, it is observed that STAMP and STOMP have a certain time complexity in determining the length of the query. On the other hand, the proposed NMP appears to be compatible with the interval of the query. Based on the results, it is observed that NMP takes 104.1 milliseconds to detect the maximum 18,000 time series-length queries.

In Figure 4b, the maximum 45,000 time series-length query is used. The result demonstrates that NMP takes 236.3 milliseconds to complete the 45,000 time series-length query. However, the contending methods STAMP and STOMP take 331.3 and 302.9 milliseconds, respectively. It is also observed that the proposed NMP rapidly increases the time after 600 time series lengths. The compared STAMP and STOMP increase the time exponentially.

Figure 4c shows that the proposed NMP takes 520.7 milliseconds to process the maximum 90,000 time series-length queries, whereas the compared algorithms STOMP and STAMP take 631.8 and 668.2 milliseconds, respectively. The proposed NMP shows constant performance in time series detection after 50,000 time series lengths. On the other hand, the compared algorithms increase the time exponentially, and the variable performance of the compared algorithms can affect the accuracy.

Figure 5a shows that the proposed NMP takes 959.2 milliseconds to process the maximum 135,000 time series-length queries, while the compared algorithms STOMP and STAMP take 1040.6 and 1074.9 milliseconds, respectively. The proposed NMP shows variable performance for time series detection. However, the proposed NMP takes less time than the compared algorithms. The compared algorithms increase the time exponentially despite having extended time series length.
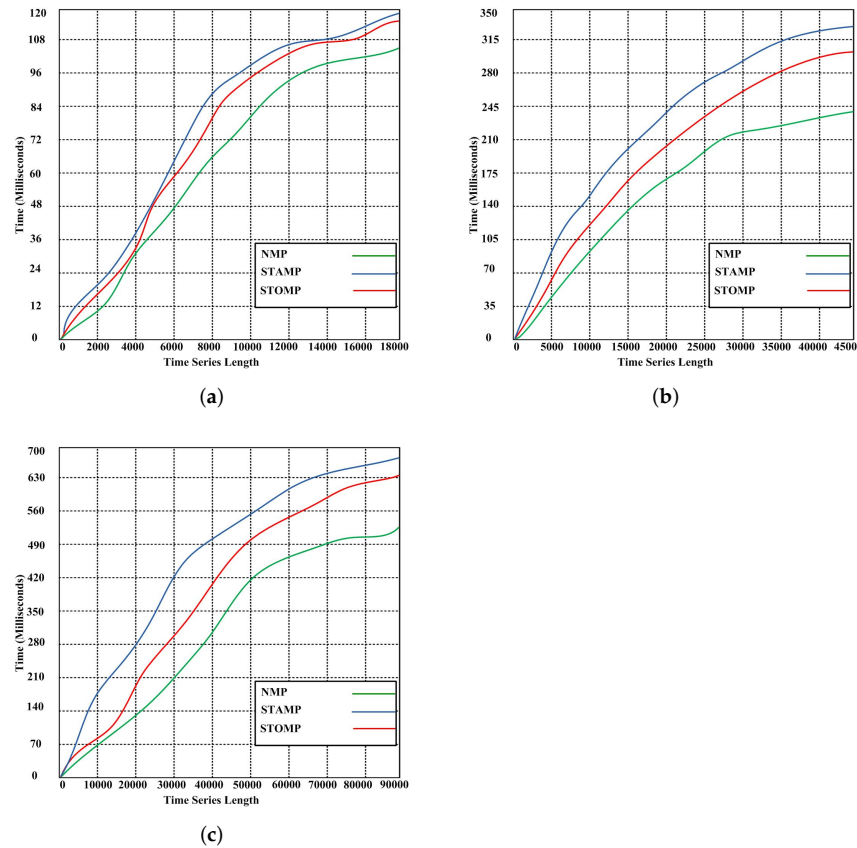
(a)

(b)

(c)

Figure 4. (**a**) Run-time execution of NMP, STAMP, and STOMP with a maximum query length of 18,000. (**b**) Run-time execution of NMP, STAMP, and STOMP with a maximum query length of 45,000. (**c**) Run-time execution of NMP, STAMP, and STOMP with a maximum query length of 90,000.

4.2.2. Accuracy

The reconstruction accuracy of the time series is assessed. The main purpose of this experiment is to compute the amount of evidence loss, which is indicated by the results for the dimensionality reduction and time series reconstruction processes. The reconstruction accuracy depends on the Euclidean interval and the dimensionality of the time series. We can calculate it as a root-mean-square deviation:

$$a_{req}(T, T') = \sqrt{\frac{\sum_{i=1}^{N}(t'_i - t_i)^2}{D_i}}, \tag{48}$$

where T is the original time series, T' is the time series reconstruction, a_{rec} is the reconstruction accuracy between T and T' and D_i is the dimensionality between time series T and T'.

The percentage of reconstructed accuracies can be seen in Figure 5b. In addition, we increase the length of the time series to 135,000 to obtain exact plotting results.

The results in Figure 5a show that all representations have good results, but the NMP algorithm gives better results than the other algorithms. We observed that by increasing the length of the data, the accuracy of the STAMP and STOMP algorithms started to decrease. When NMP shows stable accuracy, it is found to be 99.76% with a maximum time series data length of 135,000. The compared STAMP and STOMP reduce the accuracy with an increase in the length of the time series. STAMP and STOMP show 98.62% and 98.09% accuracy, respectively.

In Figure 5c, we observe that for a time series of length 180,000, the accuracy of all representations starts at 100%. After increasing the data length to 180,000, the accuracy

of STAMP drops to 98.58%. Additionally, STOMP's reconstruction accuracy decreases to 97.61% when NMP shows an almost constant accuracy of 99.5%.
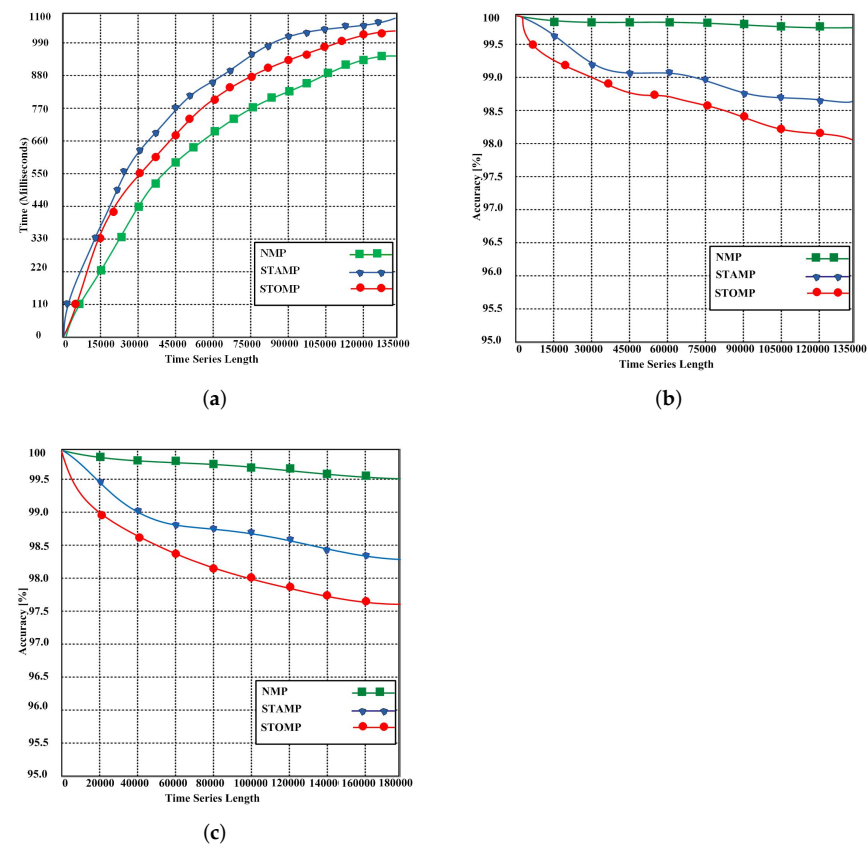

Figure 5. (**a**) Run-time execution in seconds for NMP, STAMP, and STOMP with $n = 135,000$, varying the time series length. (**b**) Accuracy of NMP, STAMP, and STOMP with a maximum of 135,000 for the time series data length. (**c**) Accuracy of NMP, STAMP, and STOMP with a maximum of 180,000 for the time series data length.

4.2.3. Detection Efficiency

SARS-CoV-2 is being updated on a daily basis. To train the data, a self-to-self transfer-learning mechanism is applied. Every month, the data were re-trained. It has been observed that the SARS-CoV-2 data set has more information as time passes. It was difficult to find information for each country. Although various countries have distinct discriminations that cause the disease to spread in different ways. The primary purpose is to establish the detection efficiency of the proposed NMP and competing algorithms: STAMP and STOMP, as well as how they determine the trend of recovered, death, and infected patients. The experiment was carried out using data from January 2020 to October 2020. Figure 6a depicts the infected patients' detection efficiency. The trend in Figure 6a demonstrates that the suggested NMP with infected patient data has an overall detection efficiency of 97.2%, whereas STAMP has an overall detection efficiency of 87.4% and STOMP has an overall detection efficiency of 87.1%.

The recovered patient data's detection efficiency is displayed in Figure 6b. The suggested NMP yields a 98.4% total detection efficiency. The competing algorithms STAMP and STOMP, on the other hand, yield total detection efficiency of 85.6% and 83.6%, respectively. The proposed NMP's detection efficiency, evaluated at 94.1% in Figure 6c using death patient data, is higher than that of the competing algorithms, which had detection efficiencies of 85.3% and 81.7% for STAMP and STOMP, respectively. According to the findings, the suggested NMP outperforms competing algorithms in terms of total detection efficiency.
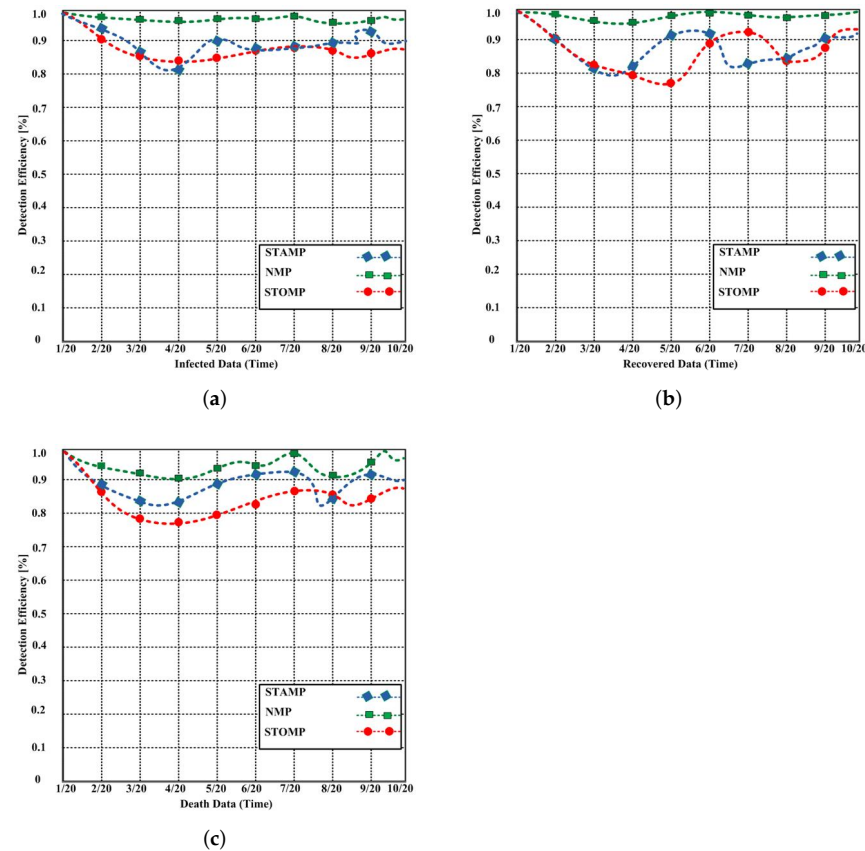
(a)

(b)

(c)

Figure 6. (**a**) Detection efficiency of the proposed NMP and contending STAMP and STOMP algorithms with patient's infected data. (**b**) Detection efficiency of the proposed NMP and contending STAMP and STOMP algorithms with patient's recovered data. (**c**) Detection efficiency of the proposed NMP and contending STAMP and STOMP algorithms with patient's death data.

5. Discussion of the Results and Limitations

The proposed NMP possesses interesting properties to meet the requirements of matrix profile indexing. It is faster than the STAMP and STOMP algorithms and occupies less space compared to the other algorithms. The proposed NMP has computational complexity O(n), whereas STOMP has complexity O(n2) and STAMP has O (n2 log n). The most important feature of the proposed NMP is that it handles queries of variable lengths. There is a very marginal chance of producing false positives or false negatives. With respect to the segmented representation, our NMP is guaranteed to produce no false negatives. However, we cannot make the same claim with respect to the original results. Because such an occurrence involves pathological conditions and has never been found in any of our studies, we do not feel that this concern is a major limitation. Nevertheless, we will briefly address an extension to our representation that would allow us to change our indexing method to ensure no false negatives with regard to the original data. A different type of samples involves 11 features that take longer to deal with. As a result, this complexity challenges the limitations of STAMP and STOMP algorithms. On the other hand, the proposed NMP leverages the feature of sorting confidence level intervals that offer data about a series in which the exact value involves a confident degree of probability. Furthermore, the data representation section can be extended from a 4-tuple to a 5-tuple, where the additional element is the residual error of approximating the rows. It is possible to adjust the distance measures stored in the index to provide a confidence bound, which is simply the sum of all the residual error terms from all of the segments that are compared. To determine the best segmented match for a query, an initial run of an indexing scheme can be used. The raw data can be indexed using this best fit, and the real Euclidean distance

between the query and the best segmented raw data can be computed. Table 2 shows the performance of the proposed and compared algorithms.

Table 2. The performance of the proposed and compared algorithms.

Algorithm	STAMP	STOMP	NMP
Time required for the similarity search with 18,000 time series	118 milliseconds	115 milliseconds	104.1 milliseconds
Time required for the similarity search with 45,000 time series	331.3 milliseconds	302.9 milliseconds	236.3 milliseconds
Time required for the similarity search with 90,000 time series	520.7 milliseconds	631.8 milliseconds	668.2 milliseconds
Time required for the similarity search with 135,000 time series	959.2 milliseconds	1040.6 milliseconds	1074.9 milliseconds
Accuracy with 135,000 time series	98.62%	98.09%	99.76%
Accuracy with 180,000 time series	98.58%	97.61%	99.5%
10 months of patient's infection data	87.4%	87.1%	97.2%
10 months of patient's recovered data	85.6%	83.6%	98.4%
10 months of patient's death data	85.3%	81.7%	94.1%

6. Conclusions and Future Work

This section describes the achieved objectives and concludes the key findings for the reader. Additionally, it offers insight for future analysis.

6.1. Conclusions

In conclusion, we propose a novel matrix profile algorithm for time series subsequence all-pairs similarity search. The proposed novel matrix profile is obtained in three phases. In the beginning, we preprocess and represent data sets by preventing time series issues such as amplitude, scaling, temporal warping, noise, and outliers. Then, we go through the distance determination process for neighboring nodes using the z-normalized Euclidean distance method. The last phase consists of the immediate neighbor detection process used to determine the values required for the matrix profile index. The NMP has good qualities such as simplicity, high speed, parallelizability, and a lack of parameters. Another advantage is its accuracy. The last expression is used to calculate the accuracy values. The accuracy of using NMP is 99.5% with a maximum time series length. Additionally, the run-time execution for the NMP algorithm is computed, and the time required for the novel matrix profile is compared to that of other existing state-of-the-art algorithms. The results show that the proposed NMP is the best choice for solving the all-pairs similarity search problem. Additionally, we show that our algorithm has implications for many current tasks, such as motif discovery, discord discovery, shapelet discovery, and semantic segmentation, and new pathways can be opened up for science, including computing various definitions of time series set differences.

6.2. Future Work

We aim to focus on the security of the proposed algorithm, particularly false positives and false negatives, for the future. Additionally, vulnerabilities and potential attacks that could affect performance will be explored. Different quality-of-service metrics will also be investigated to determine the effectiveness of the proposed algorithm.

Author Contributions: A.R., conceptualization, writing, idea proposal, methodology, and results; M.A. (Marzhan Abenova), data curation, software development, submission, and preparation; M.A. (Munif Alotaibi) and B.A., review, manuscript preparation, and visualization; H.A., S.H. and A.A. review and editing. All authors have read and agreed to this version of the manuscript.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Appendix A. Definitions

Definition A1. *A time series T is a sequence of real-valued numbers t_i:*

$$T = t_1, t_2, ..., t_n \tag{A1}$$

where Tl is the length of T.

Typically, we are interested in a time series, and we are interested not in global but local properties. A local area of a time series is referred to as a subsequence:

Definition A2. *A subsequence $T_{i,m}$ of a time series T is a continuous subset of the values from T of length m starting at position i. Formally,*

$$T_{i,m} = t_i, dt_{i+1}, ..., t_{i+m-1}, \tag{A2}$$

where i is the index of time series T, where $1 \leq i \leq m-1$, and d is the distance, where $d \ll m-1$.

Here, we can show that if we order the distance measurements for a given time series query based on the absolute values of the Z-normalized values, the average cumulative squared distance will be maximized. In other words, it is the right order in the general or in the normal case.

Definition A3. *An all-subsequences set A of a time series T is an ordered set of all possible subsequences of T obtained by sliding a window of length m across T:*

$$A = \left\{ T_{1,m}, T_{2,m}, ..., T_{n-m+1,m} \right\} \tag{A3}$$

where m is a user-defined subsequence length. We use $A[i]$ to denote $T_{i,m}$.

Definition A4. *1NN-join function: given two all-subsequences sets A and B and two subsequences $A[i]$ and $B[j]$, a 1NN-join function $\theta 1nn\ (A[i], B[j])$ is a Boolean function that returns "true" only if $B[j]$ is the nearest neighbor of $A[i]$ in set B. With the defined join function, a similarity join set can be generated by applying the similarity join operator on two input all-subsequences sets.*

Definition A5. *A self-similarity join set J_{AA} is the result of a similarity join of the set A with itself. We denote this formally as:*

$$J_{AA} = A \bowtie_{\theta 1nn} A, \tag{A4}$$

where J_{AA} is the similarity join of sets $A[i]$ and $A[j]$.

Two sequences X and Y are similar if they have long common subsequences X' and Y' such that:

$$Y' \approx aX' + b, \tag{A5}$$

where a, b are constants or coefficients that are real numbers, $a_1, ..., a_n \neq 0$ and $b_1, ..., b_n \neq 0$. The overall similarity measure maximizes the above expression over all possible transformations f, and thus, we have:

$$Sim(X, Y) = max_{allf} \left\{ Sim_f(X, Y) \right\}, \tag{A6}$$

where $Sim(X, Y)$ is the similarity between sequences X and Y; max_{allf} is the local maximum of the first-order landmarks, where $Sim_f(X, Y)$ is the similarity between the sequences:

$$X = x_1, x_2, ..., x_n, \text{ and } Y = y_1, y_2, ..., y_n.$$

Definition A6. *Similarity join set: Given all-subsequences sets A and B, a similarity join set J_{AB} of A and B is a set that contains pairs of each subsequence in A with its nearest neighbor in B:*

$$J_{AB} = \left\{ \langle A[i], B[j] \rangle | \theta_{1NN}(A[i], B[j]) \right\}, \tag{A7}$$

where J_{AB} is a similarity join set between A and B; $A[i]$, $B[j]$ are subsequences from the same all-subsequences set A. We denote this formally as:

$$J_{AB} = A \bowtie_{\theta 1NN} B, \tag{A8}$$

where $\bowtie_{\theta 1NN}$ is a 1NN-join function of sets A and B such that $J_{AB} \neq J_{BA}$.

Definition A7. *A matrix profile P of time series T is a vector of the Euclidean distances between every subsequence of T and its nearest neighbor in T. Formally,*

$$P = [min(D_1), min(D_2), ..., min(D_{n-m+1})],$$

$$P = min([MP; d]), \tag{A9}$$

where $D_i(1 \leq i \leq n - m + 1)$ is the distance profile D_i that corresponds to query $T_{i,m)}$ and time series T.

Definition A8. *A matrix profile index I of time series T is a vector of integers:*

$$I = [I_1, I_2, ..., I_{n-m+1}], \tag{A10}$$

where I is the index in $I_i = j$ if $d_{i,j} = min(D_i)$.

Definition A9. *A time series motif is the most similar subsequence pair of a time series. Formally, $T_{a,m}$ $T_{b,m}$ is a motif pair if:*

$$dis(T_{a,m} T_{b,m}) \leq dis(T_{i,m}, T_{j,m}) \in [1, 2, ..., n - m + 1], \tag{A11}$$

where $a \neq b$, $i \neq j$, and dist is a function that computes the z-normalized Euclidean distance between the input subsequences.

References

1. Li, H. Time works well: Dynamic time warping based on time weighting for time series data mining. *Inf. Sci.* **2021**, *547*, 592–608. [CrossRef]
2. Sattari, M.T.; Avram, A.; Apaydin, H.; Matei, O. Soil Temperature Estimation with Meteorological Parameters by Using Tree-Based Hybrid Data Mining Models. *Mathematics* **2020**, *8*, 1407. [CrossRef]
3. Zhang, S.Q.; Zhou, Z.H. Harmonic recurrent process for time series forecasting. In Proceedings of the European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 29 August—8 September 2020; IOS Press: Amsterdam, The Netherlands, 2020; pp. 1714–1721.
4. Soleimani, G.; Abessi, M. DLCSS: A new similarity measure for time series data mining. *Eng. Appl. Artif. Intell.* **2020**, *92*, 103664. [CrossRef]
5. Gharghabi, S.; Ding, Y.; Yeh, C.C.M.; Kamgar, K.; Ulanova, L.; Keogh, E. Matrix profile VIII: Domain agnostic online semantic segmentation at superhuman performance levels. In Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM), New Orleans, LA, USA, 18–21 November 2017; IEEE: New Orleans, LA, USA, 2017; pp. 117–126.
6. Gharghabi, S.; Yeh, C.C.M.; Ding, Y.; Ding, W.; Hibbing, P.; LaMunion, S.; Keogh, E. Domain agnostic online semantic segmentation for multi-dimensional time series. *Data Min. Knowl. Discov.* **2019**, *33*, 96–130. [CrossRef]
7. Guigou, F.; Collet, P.; Parrend, P. SCHEDA: Lightweight Euclidean-like heuristics for anomaly detection in periodic time series. *Appl. Soft Comput.* **2019**, *82*, 105594. [CrossRef]

8. Hu, M.; Feng, X.; Ji, Z.; Yan, K.; Zhou, S. A novel computational approach for discord search with local recurrence rates in multivariate time series. *Inf. Sci.* **2019**, *477*, 220–233. [CrossRef]
9. Zhou, Y.; Ren, H.; Li, Z.; Pedrycz, W. An anomaly detection framework for time series data: An interval-based approach. *Knowl.-Based Syst.* **2021**, *228*, 107153. [CrossRef]
10. Li, J.; Izakian, H.; Pedrycz, W.; Jamal, I. Clustering-based anomaly detection in multivariate time series data. *Appl. Soft Comput.* **2021**, *100*, 106919. [CrossRef]
11. Crnkić, A.; Ivanović, I.; Jaćimović, V.; Mijajlović, N. Swarms on the 3-sphere for online clustering of multivariate time series and data streams. *Future Gener. Comput. Syst.* **2020**, *112*, 11–17. [CrossRef]
12. Yu, C.; Luo, L.; Chan, L.L.H.; Rakthanmanon, T.; Nutanong, S. A fast LSH-based similarity search method for multivariate time series. *Inf. Sci.* **2019**, *476*, 337–356. [CrossRef]
13. Yeh, C.C.M.; Zhu, Y.; Ulanova, L.; Begum, N.; Ding, Y.; Dau, H.A.; Silva, D.F.; Mueen, A.; Keogh, E. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; IEEE: Barcelona, Spain, 2016; pp. 1317–1322.
14. Xiang, L.; Leng, P.; Zhang, J.; Luo, K.; Yang, Z.; Nai, W. Principal Component Analysis Based on Artificial Fish Swarm with T-Distribution Parameters. In Proceedings of the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Xi'an China, 15–17 October 2021; IEEE: Xi'an China, 2021; Volume 5, pp. 2373–2377.
15. Alexander Scarlat MD "Time Series with Anomalies". Available online: https://www.kaggle.com/datasets/drscarlat/time-series (accessed on 10 March 2022).
16. Zhu, Y.; Zimmerman, Z.; Senobari, N.S.; Yeh, C.C.M.; Funning, G.; Mueen, A.; Brisk, P.; Keogh, E. Matrix profile II: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In Proceedings of the 2016 IEEE 16th International Conference on data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; IEEE: Barcelona, Spain, 2016; pp. 739–748.
17. Gowanlock, M.; Karsin, B. Accelerating the similarity self-join using the GPU. *J. Parallel Distrib. Comput.* **2019**, *133*, 107–123. [CrossRef]
18. Zhao, H.; Wang, Y.; Duan, J.; Huang, C.; Cao, D.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; Zhang, Q. Multivariate time-series anomaly detection via graph attention network. In Proceedings of the 2020 IEEE International Conference on Data Mining, Sorrento, Italy, 17–20 November 2020; pp. 841–850.
19. Liu, D.; Li, J.; Yuan, Q. A spectral grouping and attention-driven residual dense network for hyperspectral image super-resolution. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 7711–7725. [CrossRef]
20. Wan, P.; Davis, R.A. Goodness-of-fit testing for time series models via distance covariance. *J. Econom.* **2020**, *227*, 4–24. [CrossRef]
21. Kalmykov, L.V.; Kalmykov, V.L. A solution to the dilemmalimiting similarity vs. limiting dissimilarity'by a method of transparent artificial intelligence. *Chaos Solitons Fractals* **2021**, *146*, 110814. [CrossRef]
22. Tsuchiyama, A.; Nakajima, J. Diversity of deep earthquakes with waveform similarity. *Phys. Earth Planet. Inter.* **2021**, *314*, 106695. [CrossRef]
23. Rubinstein, B. A fast noise filtering algorithm for time series prediction using recurrent neural networks. *arXiv* **2020**, arXiv:2007.08063.
24. Wang, D.; Gao, X.; Wang, X.; He, L. Label consistent matrix factorization hashing for large-scale cross-modal similarity search. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 2466–2479. [CrossRef]
25. Johnson, J.; Douze, M.; Jégou, H. Billion-scale similarity search with gpus. *IEEE Trans. Big Data* **2019**, *7*, 535–547. [CrossRef]
26. Ma, L.; Gu, X.; Wang, B. Correction of outliers in temperature time series based on sliding window prediction in meteorological sensor network. *Information* **2017**, *8*, 60. [CrossRef]
27. Cai, X.; Xu, T.; Yi, J.; Huang, J.; Rajasekaran, S. Dtwnet: A dynamic time warping network. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 11640–11650.
28. Pelletier, C.; Webb, G.I.; Petitjean, F. Temporal convolutional neural network for the classification of satellite image time series. *Remote Sens.* **2019**, *11*, 523. [CrossRef]
29. Smyl, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *Int. J. Forecast.* **2020**, *36*, 75–85. [CrossRef]
30. Hong, J.Y.; Park, S.H.; Baek, J.G. SSDTW: Shape segment dynamic time warping. *Expert Syst. Appl.* **2020**, *150*, 113291. [CrossRef]
31. Rubinstein, A.; Song, Z. Reducing approximate longest common subsequence to approximate edit distance. In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Lake City, UT, USA, 5–8 January 2020; pp. 1591–1600.
32. Vishwakarma, G.K.; Paul, C.; Elsawah, A.M. An algorithm for outlier detection in a time series model using backpropagation neural network. *J. King Saud-Univ.-Sci.* **2020**, *32*, 3328–3336. [CrossRef]
33. Linardi, M.; Zhu, Y.; Palpanas, T.; Keogh, E. Matrix profile X: VALMOD-scalable discovery of variable-length motifs in data series. In Proceedings of the 2018 International Conference on Management of Data, Houston, TX, USA, 10–15 June 2018; pp. 1053–1066.
34. Dong, E.; Du, H.; Gardner, L. An interactive web-based dashboard to track COVID-19 in real time. *Lancet Infect. Dis.* **2020**, *20*, 533–534. [CrossRef]