IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 31 December 2022, accepted 16 January 2023, date of publication 23 January 2023, date of current version 26 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3239212

APPLIED RESEARCH

High Performance Time Series Anomaly Detection Using Brain Inspired Cortical Coding Method

MERIC YUCEL[1,2], AHMET SERTBAS[2], AND BURAK BERK USTUNDAG[3], (Member, IEEE)
[1]National Software Certification Research Center, Istanbul Technical University, 34467 İstanbul, Turkey
[2]Department of Computer Engineering, Istanbul University-Cerrahpasa, 34320 İstanbul, Turkey
[3]Department of Computer Engineering, Istanbul Technical University, 34467 İstanbul, Turkey

Corresponding author: Meric Yucel (yucelme@itu.edu.tr)

This work was supported by the Research Project "Platform Development for Neuromorphic Computing and Next Generation Programming" funded by the National Software Certification Research Center, Istanbul Technical University.

ABSTRACT Accurate and automated anomaly detection in time series data sets has an increasingly important role in a wide range of applications. Inspired by coding in the cortical networks of the brain, here we introduce a novel approach for high performance real-time anomaly detection. Cortical coding method is adaptive and dynamic, consisting of self-organized networks. In the cortical coding network introduced herein, the morphological structuring is driven by a brain inspired feature extraction strategy that aims the minimization of the signal energy dissipation while increasing the information entropy of the system. We combine the cortical coding network with transform coding and multi resolution analysis for anomaly detection. As we demonstrate here, the new coding methodology provides high computational efficiency in addition to scalability with respect to target accuracy compared to the traditional clustering algorithms. A wide variety of data sets are used to demonstrate time series anomaly detection performance. In a preliminary work presented here, we detected 77.6% of the present anomalies correctly, using the same hyperparameters for every stage of the method. The results are compared with several clustering algorithms such as K-means and its variants mini-batch K-means, sequential K-means and finally with hierarchical agglomerative clustering. Additionally, the performance of all the clustering methods are compared by memorizing all input data set without performing any clustering. The cortical coding method has shown the best performance compared to the other methods. From the results achieved so far, it appears that there is still a significant room for improvement of the success rate by, specifically, performing hyperparameter and filter optimization according to the characteristics of data sets and using a more advanced fusion model at the output layer. Low time and space complexity, high generalization performance, suitability to real-time anomaly detection, and in-memory processing compatibility are distinct advantages of the cortical coding method in a variety of anomaly detection problems, such as predictive maintenance, cybersecurity, telemedicine, risk management, and transportation safety.

INDEX TERMS Anomaly detection, brain inspired cortical coding network, clustering, cybersecurity, discrete wavelet packet transform, entropy maximization, hierarchical agglomerative clustering, k-means, mini-batch k-means, sequential k-means, time series data.

I. INTRODUCTION

Accurate and early detection of anomalies in time series data is pivotal in a wide range of applications including cybersecurity [1], smart power grids [2], intrusion [3], [4] and fraud detection [5], medical and public health [6], and the sensor networks [7], [8], [9]. The majority of the conventional anomaly detection methods are case specific. Although AI-based anomaly detection improvements partially compensate for this generalization drawback, their

The associate editor coordinating the review of this manuscript and approving it for publication was Zhigao Zheng.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

real-time applicability is limited due to the higher space and time complexity rates [9], [10]. Here we introduce a brain inspired cortical coding method that mimics the energy-entropy relationship of the neural network formation in the brain [11] and apply it to demonstrate time series anomaly detection in various data domains. The proposed cortical coding method is self organized, dynamic and adaptive. The key advantages of the proposed method are good generalization performance, low time complexity and the compliance with online learning [11].

In a generalized technical sense, an anomaly is the deviation from normal which does not follow the collective common pattern and, hence, it can be distinguished from the whole data set [9]. In a time series data, such as a cyclic, seasonal, or common collective trend, a data point that deviates from the pattern with a statistically significant level is called an anomaly [12], [13]. Anomaly detection, also often referred as outlier [14] or novelty detection [15], is the process of finding unexpected or abnormal patterns in data sets [16].

In time series, an anomaly can occur as point, contextual, or collective abnormality [9], [17]. Point anomaly is a single point that deviates statistically from the rest of the data and can mostly be easily detected by simple statistical detection methods [7], [9]. Contextual anomalies refer to those that do not deviate from the rest of the data, except within a context [18]. Collective anomalies, on the other hand, refer to the cases where a collection or a group of data exist that have anomalous characteristics with respect to the rest of the data [9]. In time series, anomalies experienced are mostly contextual or collective [19], and are harder to detect, often needing more elaborate approaches to detect. Overall, therefore, anomaly detection in time series is a challenging task in distinguishing what is normal or abnormal, especially at the boundaries if the data is noisy or scarce. Additionally, malicious activities often aim to hide abnormalities to avoid detection through a never ending obfuscation/deobfuscation game between malware developers and cybersecurity experts. Since the systems are dynamic, the definition of normal and abnormal data may change in time and, therefore, a well designed outlier detection algorithm should be adaptable to the dynamic process, although the lack of labelled data or unbalanced normal or abnormal data may make this difficult to achieve. Due to these challenges, most anomaly detection algorithms are designed to perform for a specific application and, hence, the success rate tend to decrease when applied for a different application domain [9], [20], [21], [22]. The ultimate goal, therefore, is to develop a more generalized algorithm for reliable detection of various kinds of anomalies in a wide range of real-life problems.

A time series $X_t, t \in T$ is a collection of data samples recorded and indexed by over the period time T, which could be discrete or continuous. There are numerous methods developed for anomaly detection in time series, the classification of which may differ depending on the type of data and the kind of anomaly that may exist [14]. Among the many classifications [23], [24], [25], data driven methods are mainly categorized as machine learning (ML) and statistical approaches [26]. Following the recent developments, the ML category is further divided as conventional and deep learning methods [19]. In both cases, the characteristics of univariate time series data, e.g., being stationary, periodic, and having seasonality or trend [19], effect the success of the model. Here, statistical conventional methods such as Seasonal Auto-regressive Integrated Moving Average (SARIMA) and Seasonal Trend Decomposition (STL) are generally used for periodic and stationary time series [19], [27]. Conventional ML methods in anomaly detection are broadly researched area [28], [29]. Some conventional ML approaches, e.g., cluster-based methods, have mostly the same challenges, i.e., abnormal activities are modeled in cluster-based approaches. The challenging issue here is defining the optimized cluster size and deciding whether to classify new instances as abnormal [30]. With the recent developments in artificial intelligence, deep learning methods are becoming more popular. Various methods are applied to the domains of different implementations [31], [32] and they have been comprehensively studied [33], [34]. Although many specialized deep learning methods in anomaly detection have shown promising success when compared to the conventional methods, there are still challenging problems [35].

The implementations of the presently used methods in a time series anomaly detection, in general, are highly inconsistent in different applications as the relationship among them in terms of their performance in a given set of conditions are not known which, in turn, may lead to incorrect use of a methodology for a given data resulting in low predictability or irrelevant results. This is especially important when variety of methodologies are applied for popular benchmark data sets such as Numenta, Yahoo, and NASA, to achieve an average predictability score, which may then create an illusion of progress [36].

The cortical coding method introduced here addresses these difficulties and overcomes the limitations in anomaly detection in time series data in a wide variety of cases, as we demonstrate by using commonly employed data set. The major characteristics of the cortical coding method [37] and the contributions of the work can be summarized as follows. The method:

- Is self-organized, adaptive, and dynamic and, therefore, suitable for unsupervised online anomaly detection;
- Has low time and space complexity (time complexity is $O(ndlogm)$, where n is the number of vectors, d is the dimension, and m is the maximum number of progeny nodes of a cortex node in the evolving cortical coding tree, and since $m \ll n$, the time complexity can be considered as $O(nd)$, the same as space complexity);
- Requires less hyperparameter tuning and, being a clustering based method, has excellent generalization performance;

- Acts like a feature extractor from a highly complex variety of data since the cortical layers of the brain inspired cortical network is self-generating;
- Has the best performance compared with several popular clustering methods;
- Is compatible with in-memory processing and, thereby, takes advantage of neuromorphic computing.

In a preliminary demonstration using all the 250 Anomaly Benchmark data sets, provided by the University of California, Riverside (UCR) [38], the cortical coding methodology scored 77.6% true positive in anomaly detection without the need for optimization. In the next section, we present the working principles of the cortical coding method, demonstrate its performance using the UCR benchmark data set and compare its success at 95 percentile in a worldwide competition [39]. To demonstrate the performance of the cortical coding method, we compared it with that of the frequently used, representative, K-means, mini-batch K-means, sequential K-means, hierarchical agglomerative clustering and memorizing all the input data set methods while keeping all the prepossessing step the same. As we demonstrate below in detail, the self-generation of the cortical layers of the brain inspired cortical network acts like a feature extractor from a highly complex variety of data following the principle of entropy maximization while minimizing the energy. The main advantage of the methodology is its low time and space complexity, while being case non-specific. These aspects of the newly introduced methodology derived from a one-to-one comparison of the cortical coding approach with several frequently used anomaly detection methods.

II. METHODOLOGY

The cortical coding network is a self-organized, adaptive, hierarchical clustering based method. It has an n-array tree-like structure and is a combination of transform coding and clustering [11]. The new method is inspired by the sensory cortical network found in biological brain, whose structuring is accomplished based on the rules of thermodynamics. The key principle in this cortical model is to structure a decoder network that converts input signals with an initially unknown probabilistic distribution, into an output that converges to a uniform distribution set of features (see Fig. 1). The features having a uniform distribution of information is necessary for optimal coding and highly efficient processing of the information [40].

The cortical coding network introduced here has three key aspects, each of which is inspired from human brain. Firstly, it is assumed, from the start, that stem cells are present and homogeneously distributed throughout the computational space, i.e., an imagined synthetic brain. The consequence of the presence of stem cells is that any initially active given neuron has a set of distributed spines which are candidates to be potential future synaptic connections, and each representing possible feature of the input signal that could be electrically conducted or terminated. Depending on the input signal, the spines representing the minimum feature distance transfer the higher portion of the signal as an electromagnetic force that stimulates the stem cells down its information gradient, while retaining the rest of the energy to accumulate within the imagined ''soma'' of the neuron, analogous to the neural network formation in the brain. In other words, if the accumulated energy exceeds the initially set threshold value within the stimulated spine direction, then it is converted into a new synaptic connection by transforming the stem cell into a new neuron depending on the progenitor process at work.

TABLE 1. Nodes of the cortical coding network.

	Cortex Node	Spine Node
Parent Node	Yes (Cortex)	Yes (Cortex)
Progeny Node	Yes (Cortex and/or Spine)	No
Node Value	Yes	Yes
Range Parameter	Yes	Yes
Pass count Parameter	Yes	Yes
Maturity (Dissipated energy)	No	Yes

Secondly, a simple rule is used for the morphological structuring of the neurons and their connections, i.e., the restructuring happens by the neuron that yields the maximum thermodynamic and information entropy simultaneously. This condition is achieved by representing the transferred signal as an energy dissipation process in terms of a specific level of heat. In this scenario, the dissipated energy is defined as the square power of the error between the coefficient in cortex node and the input signal.

Thirdly, inspired by the synaptic plasticity in the brain, the adaptation of the well-trained node is assumed to be slower than that of the less trained node, which is then controlled with a weight parameter in cortical coding network. While plasticity in general, is referred to as the brain's ability to adapt to the new information, the synaptic plasticity is the activity dependent change of the strength or efficiency of the connections among the neurons [41], [42]. Fig. 2 summarizes the inspiration from brain, the development of the cortical coding network, and the formation of its general structure.

In the cortical coding tree, there are two types of nodes. The first type is the cortex node, i.e., the main node of the cortical coding tree. The second type is the spine node. These nodes are used only during the training phase or when the continuous learning phase is active. When training is completed or continuous learning is discarded, spine nodes are ignored and not included in the generated codebook. The spine nodes are the candidates that evolve to become cortex nodes, i.e., a well-trained spine node evolves into a cortex node. The detailed explanation of cortical coding network and its performance can be found in [11] and [37]. There are also other differences between cortex and spine nodes that define their structural characteristics and functions. The differences between the two types of nodes are presented in Table 1.

As noted from Table 1, all nodes have values, range parameters, and pass count parameters. The node values are

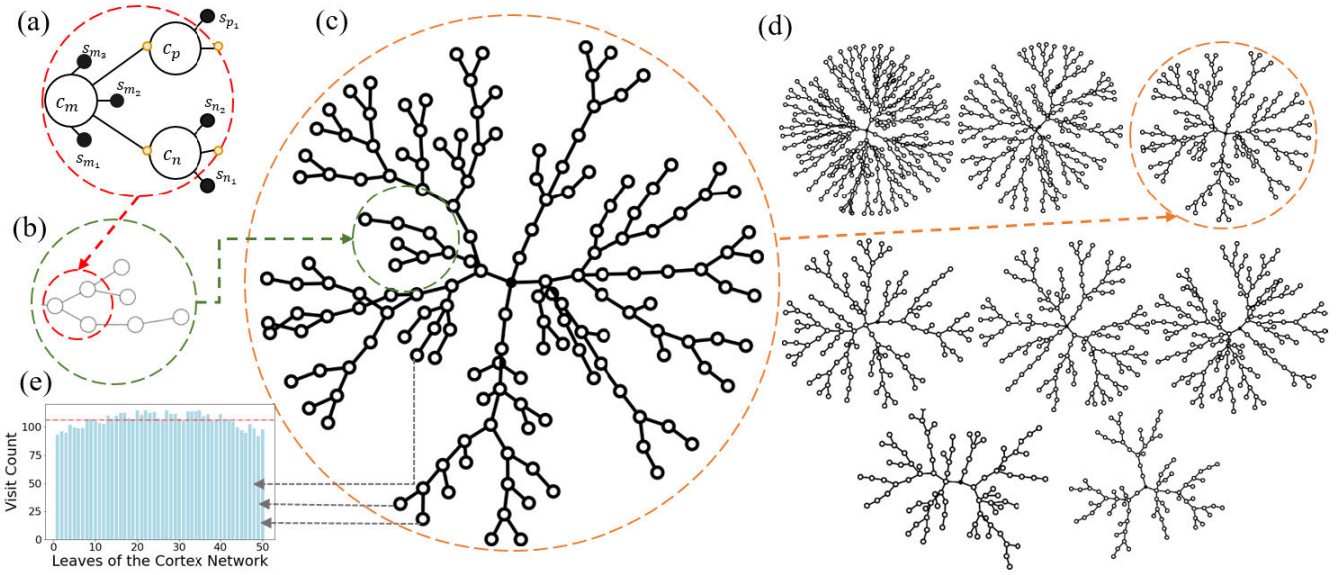


FIGURE 1. The cortical coding network for multiple frequency bands. (a) Detailed structure of the nodes; the label *c* indicates cortex nodes and *s* are the spine nodes. (b) A wider vision of cortex nodes (c) One frequency band specialized cortical coding network, (d) Cortical coding networks of eight frequency bands. (e) Histogram of the leaves of the cortical coding network, specialized in one frequency band. Red line indicates the maximum entropy, where visiting all nodes are equally probable.

scalar, where the input data are stored. The range parameter of a given node identifies the effective covering range of that node. This parameter is limited by two values, which set the boundaries for maximum and minimum, r_{init} and r_{limit}, respectively. The range parameter is, therefore, used for determining the active range for the related node. It is initialized with r_{init} value and, as the node is trained, narrows down until it reaches r_{limit}. Narrowing down the range parameter helps to generate frequent nodes where the input data are dense. Hence, the process helps to maximize information entropy and increases clustering performance.

The pass count parameter is inspired from synaptic plasticity in the brain; it simply counts how many times a node is trained or triggered. A well-trained node adapts in a slower pace, i.e., the changes in the node value is slower. Conversely, if a node is less trained, then the change in its node value is faster. The main difference between the cortex and the spine nodes are that the former can only have progeny nodes, so all the parent nodes in the cortical coding network are the cortex nodes. On the other hand, the spine nodes have maturity parameter that controls their evolution into cortex nodes. More specifically, if the cumulative dissipated energy exceeds over a predetermined threshold value, then the spine node evolves into cortex node. The detailed steps in the network formation are described in Fig. 2.

A. TRAINING AND DEVELOPMENT OF THE CORTICAL NETWORK

Certain aspects of the formation of the cortical coding network are inspired by the neural network formation in the brain. Specifically, at the initial stage, the cortical coding tree has only a root node before any training resumes. While the cortical coding tree is fed by the input data, the network grows with the formation of new branches and nodes following a set of specific rules. The consequence of the rules controlling the formation and the growth of the network is towards maximizing the information entropy while minimizing the dissipated energy, i.e., having equally probable features at the leaves of the tree. Here, the depth of the cortical coding tree is predetermined and is equal to the length of the input data vector.

As described earlier, each node, whether cortex or spine, in the cortical coding network has three parameters; a node value, effective covering range, and pass count. The node value is a scalar unit, corresponding to the worth of the input data; the effective covering identifies the operating values of the node within a given range; the pass count parameter identifies how well the node is trained or updated. Based on these parameters, there are two conditions by which a cortex node is triggered:

- The value of the cortex node has the minimum distance to the incoming data among all the sibling nodes,
- The value of the incoming data is within the covering range of the cortex node.

If these two conditions are satisfied, then the node is triggered and its value is improved according to the incoming data, minimizing the dissipated energy, by following (1):

$$\hat{c} = c + \frac{(1-k)(x_i - c)}{(w \times l + 1)^n} \tag{1}$$

where, $\hat{c}$ is the updated node value, c is the value of the node before it is triggered, k is the adaption control coefficient ($0 < k < 1$), x_i is the value of the incoming data, w is the pass

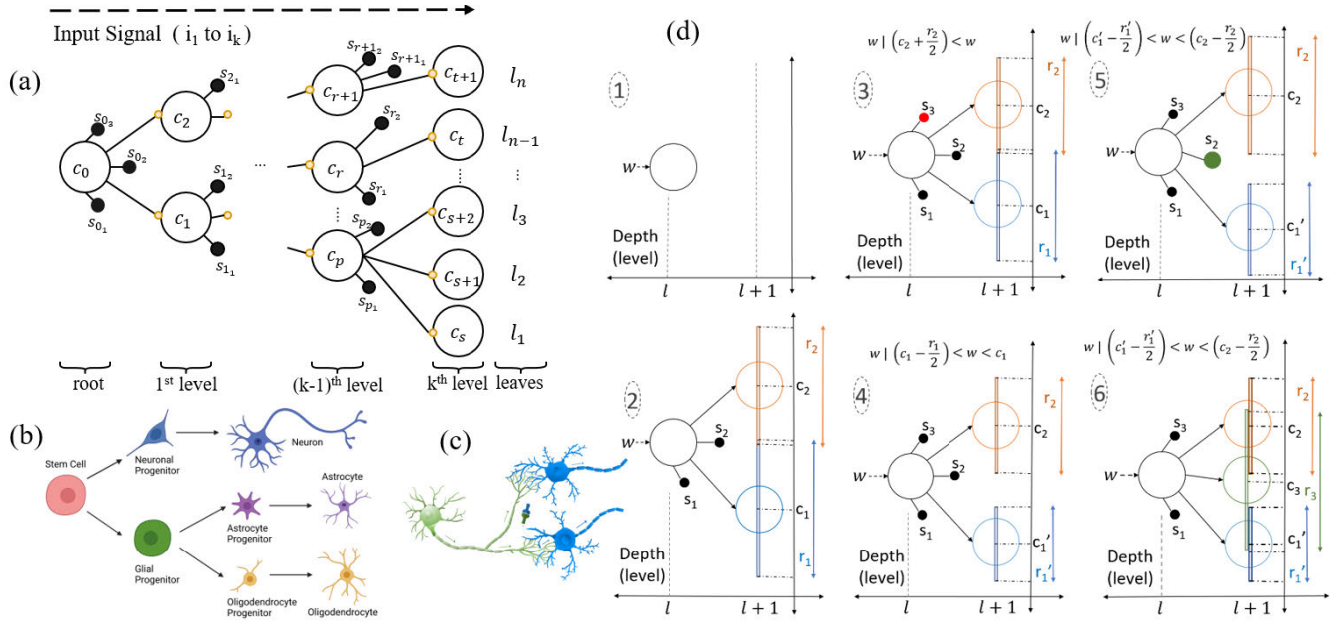


FIGURE 2. An overview of the cortical network and its development. (a) k-level cortical coding network and the nodes. (b) Progenitor cells evolving into neuron cells in the biological brain. (c) An example of connections among neurons. (d) An example of node and cortical coding network development depending on the input values (w), insets 1 through 6, respectively.

count parameter (weight), l is the level coefficient, and n is the power of the pass count control ($0.5 < n < 1$).

The pass count parameter, w, represents the maturity of the node and, if the node is well-trained, i.e., w is large, then the adaptation is slower. The l parameter is added to the equation to differentiate the adaptation speed between the levels of the cortical coding tree. Since the cortical coding tree has an n-array hierarchy, the probability of nodes to be triggered at a high level are less than those at low levels. This means that the high-level nodes are expected to be less trained than the low-level ones. For maintaining this difference throughout the network formation, l parameter is added to the equation. Therefore, if l is large, the adaptation is slower, and vice versa. The k parameter, on the other hand, controls the adaptation speed depending on the newly coming vs the historical values. Hence, the higher values of k give more importance to the history than the newly incoming input data, meaning that it slows down the adaptation. Conversely, smaller k values speed up the adaptation by giving more importance to the more recent incoming data rather than historical ones.

The minimum energy dissipation is defined in the cortical coding network as the square power of the distance between incoming data and its closest node. When the node is triggered, it updates its value to incoming data and the dissipation energy is reduced. Here, the range parameter controls the effective dynamic range of the nodes in the cortical coding tree. It starts with a predetermined value r_{init} and, as the node is trained, the range of the node shrinks. By narrowing down the range parameter, we aim to have more nodes visited with an equal probability thereby increasing the information entropy. This is because narrower range parameters increase the probability of having more neighboring nodes to be generated. Therefore, more number of nodes are generated where the input data is dense, and less number of nodes are generated where the input data is less frequent. There is a limiting value of the range parameter, r_{limit}, which sets the minimum value of the node. The range parameter is given by (2);

$$r = \begin{cases} \dfrac{r_{init}}{w^k \times l}, & r > r_{limit} \\ r_{limit}, & r \leq r_{limit} \end{cases} \tag{2}$$

where, w is the pass count parameter, l is the level coefficient (whose value is equal to the level of the node in the cortical coding tree), and k is the power of pass count to control the speed of narrowing down process. The value of the parameter k is chosen 0.5 for this study. As described earlier, by updating the node value according to (1), the dissipation energy is minimized. On the other hand, narrowing down the range parameter helps to increase the information entropy at each level of the cortical coding tree since the increasing probability of the generation of new nodes in areas where input data are denser. The overarching goal here is to reach the maximum entropy at the leaves of the cortical coding tree i.e., to have equal probable features at the leaves. Increasing the entropy at each level also increases the overall entropy at the leaves of the cortical coding tree. More detailed mathematical background of the entropy maximization in cortical coding network can be found in [11] and [37].

When the two conditions for cortex node update mentioned above are not satisfied, then the same rules are applied for

the spine nodes in the cortical coding tree; this means that the closest spine node is found and checked if the input value is in the range of the selected spine node. If it is not in the range, then a new spine node is generated with the incoming data value at that point. If it is within the range, then the spine node is updated, similar to the cortex node, according to (1) and (2). Different than the cortex node, the maturity of the spine node is evaluated after each update. If the spine node is well-trained so that its maturity value exceeds a certain predetermined threshold, then it evolves into a cortex node. The maturity of the spine node is controlled with a first order Infinite Impulse Filter (IIR), given in (3).

$$y[n] = y[n-1] + k_l x[n] \tag{3}$$

where, $y[n]$ is the maturity level, $x[n]$ is the dissipated energy, and k_l is the level coefficient to arrange the speed of evolution in different levels of the cortical coding tree. The pseudo code of the cortical coding method of a frame is presented in Appendix A.

B. APPROACH

The anomaly detection method introduced here may be considered a hybrid methodology, since it is a combination of transform coding, multi resolution analysis and cluster-based approach. Here, the input signal is divided into k frequency bands, each of which is resampled according to the frequency components within the bands. The skip parameter s is used for resampling and the highest frequency band is sampled with $s = 1$, and the lowest frequency band is skipped $s = 2^{k-1}$. Each filtered band is slid by 1 and Discrete Wavelet Packet Transform (DWPT) is carried out for every 8 samples. The window size is chosen as 8 to optimize the depth of the cortical coding tree and its complexity. Input signal from the data set is trained for each cortical coding network dedicated to each frequency band. The general schematics of preprocessing and training of the cortical coding-based anomaly detection approach is presented in Fig. 3.

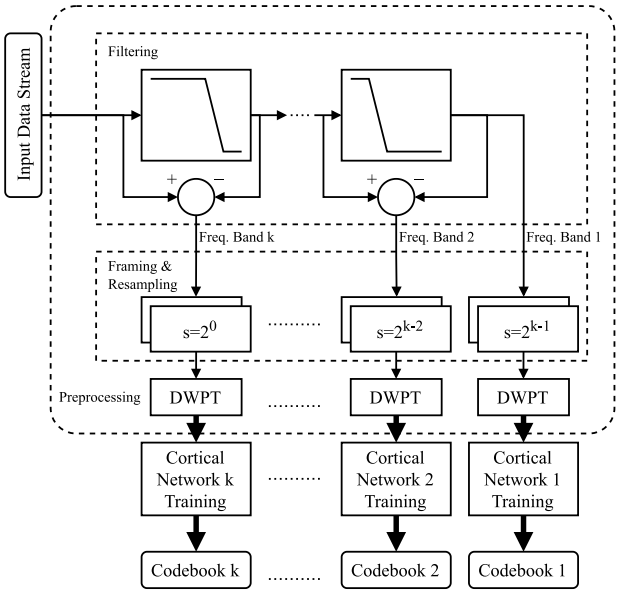


FIGURE 3. A schematic of preprocessing and training of the cortical coding method-based anomaly detection. The input signal is filtered with a sequence of low pass Butterworth filter and divided into k frequency bands. Each band is resampled and framed according to the frequency characteristics. Then, each framed window is transformed with DWPT. The wavelet coefficients are trained in the cortical coding tree. As a result, k cortical coding trees, specialized into k different frequency bands, are trained.

1) FILTERING

The input signal is filtered with a sequence of low pass filters. More specifically, decomposition into frequency bands is performed through a sequence of cascaded low pass Butterworth filters, which are designed to have smooth pass band [43]. The 3rd order Butterworth filters are used to divide input data set into eight frequency bands. The data in UCR Anomaly Benchmark set are temporal data without time labels. We, therefore, decided to set a sampling frequency for all data and chose the frequency intervals accordingly. Here, the sampling frequency is chosen as 64 kHz and the frequency bands intervals are set from high to low order as (16000, 32000), (8000, 16000), (4000, 8000), (2000, 4000), (1000, 2000), (500, 1000), (250, 500) and (0, 250).

2) FRAMING AND RESAMPLING

In this study, the frame window length is chosen to be 8 samples wide. Each frequency band is resampled according to the characteristics of the signals captured within the band by arranging the skip parameter. The skip length is arranged from the highest to the lowest frequency bands as $s = 2^k$, where $k = 0, 1, \ldots, b-1$, where b is the number of bands. The sliding parameter determines the sample difference of starting points of consecutive windows. Depending on the types of anomaly, there might be significantly narrow anomaly periods, for example, in the case of point anomalies. In order not to miss any useful information from the incoming data, the sliding parameter is set to 1 for each frequency band. In Fig. 4, an example is presented to demonstrate sliding and skipping processes for a given frequency band.

3) DISCRETE WAVELET PACKET TRANSFORM

Discrete Wavelet Transform (DWT) is a powerful transformation method that provides both frequency and localization analyses with different scales in data [44]. In the case of DWPT, unlike in DWT, the frequency division is carried on high frequency components (details) as well and, therefore, it provides more detailed frequency analyses. Since the cortical coding tree is hierarchical, it is more suitable to use DWPT rather than DWT. A schematic of three-layer DWPT is presented in the Fig. 5.

4) CORTICAL CODING NETWORK TRAINING

The input signals of the cortical coding network are the wavelet coefficients of the filtered, framed, and resampled signals in each band. At the start of the training, there is only the root node at level "0" in each band. The cortical coding

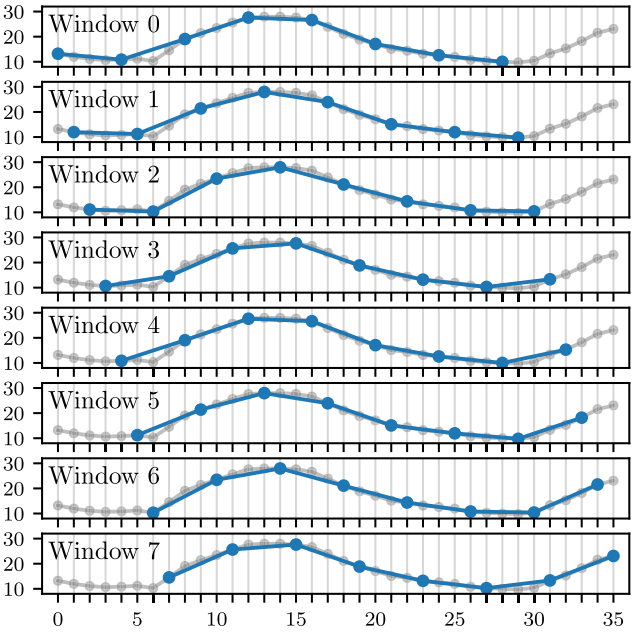


FIGURE 4. An example of sampling from the input data for the third highest frequency band. The highlighted blue dots show the sampled points and the grey points represents the input data. Here, the skipping parameter is 4 while the sliding parameter has been selected as 1.

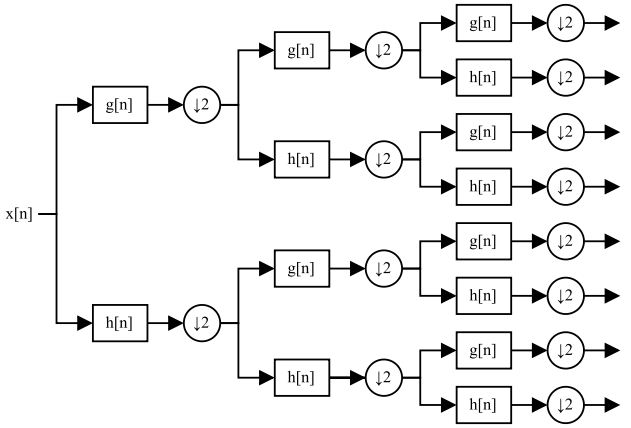


FIGURE 5. A three-layer DWPT decomposition.

network is trained with the rules that were discussed above in Section II-A. Training period of the data set is read for each band. Depending on the range of the values in the training period of the relative band, the parameters r_{limit} and r_{init} are set as defined in Equations (4) and (5)

$$r_{init_b} = \frac{max(T_b) - min(T_b)}{R_1} \tag{4}$$

$$r_{limit_b} = \frac{r_{init_b}}{R_2} \tag{5}$$

where, R_1 and R_2 are the sensitivity parameters that control the initial ranges and the limits. The value of R_1 is used to set the r_{init} parameter in relation to the range of the input data. The range of the data set in the training period (T) is found first and then the value of r_{init} is set accordingly.

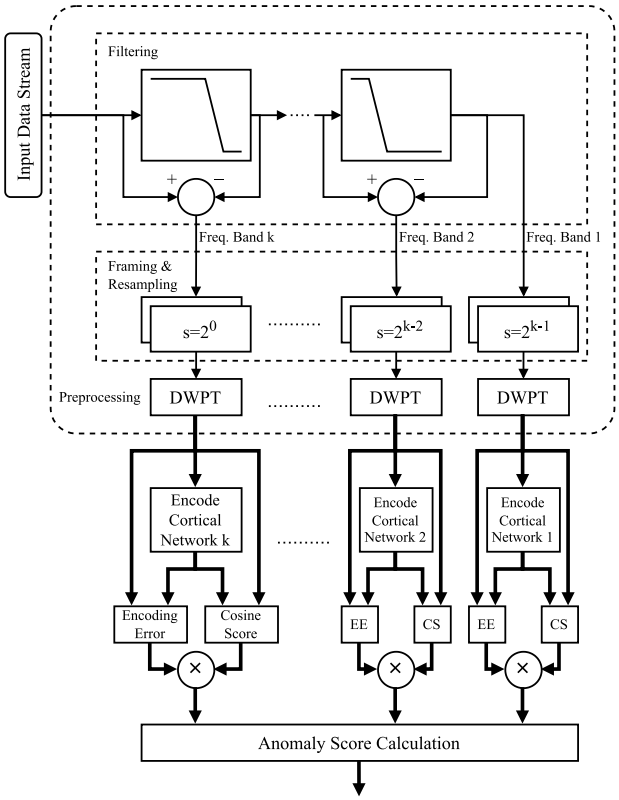


FIGURE 6. A representation showing the encoding of the trained cortical coding trees and the anomaly score calculation.

A larger r_{init} value leads to more robustness in the cortical coding network. A smaller r_{limit} parameter, on the other hand, tends to create higher number of nodes in the cortical coding network and, therefore, decreases the encoding error. Creating more nodes, however, increases the total training time. A high value of R_2 increases the ratio between r_{init} and r_{limit} and, therefore, helps in forming well-trained signals and ignores the less frequent and noisy signals in generating nodes in the network. Depending on the application type, various sensitivity parameters could be used for initializing the range parameters. The values of R_1 are set to 10 and R_2 is set to 15 for the cases used in this work. The training for each cortical coding network is repeated 8 times to construct the network so that the training period is learned and can be fully encoded in the tree.

5) ANOMALY PREDICTION

A codebook is generated for each band upon the train period is completed. The generated codebook rows are the paths from the root to the leaves of the cortical coding networks. The input signal in the test data set is preprocessed. The preprocessed signals are encoded in the cortical coding trees and compared with the generated codebooks. For each window in a given band, the closest vector in the relevant band's generated codebook is found and the encoding errors are calculated between the input signals and the encoded path in the network. The encoding error (e) is the Euclidean

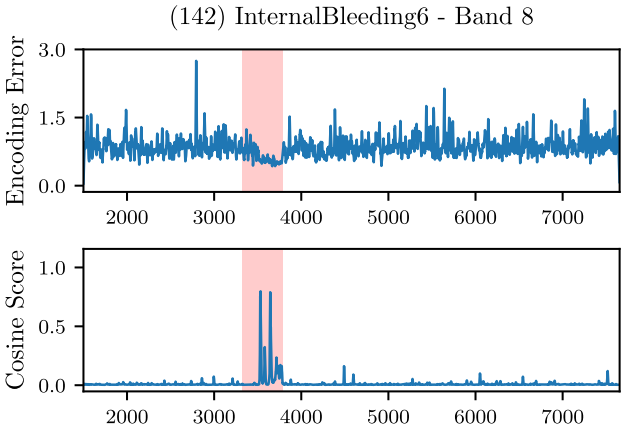


FIGURE 7. An example where cosine score finds the anomaly, metrics for data set 142, band 8. The pink area is the anomaly window. Since the amplitude of the anomaly is small, the encoding error is also small; therefore, using only the encoding error most likely does not lead to finding the correct anomaly. The cosine score, however, can find the true anomaly since it measures vectoral similarity, which is independent from the amplitude of the signal.

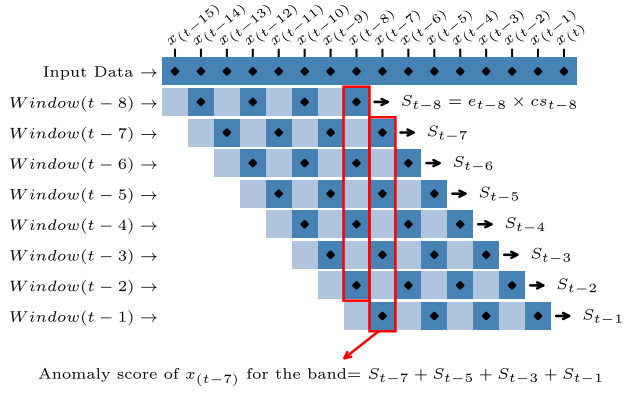


FIGURE 8. Anomaly score calculation using the overlapping windows. For convenience, the figure is drawn for window size = 4, skip size = 2, and 16 samples of input data. The dotted squares indicate the presence of samples while the empty squares indicate that these samples are skipped, not used, according to the skip function. Each resampled point uses the anomaly scores of all the windows with data. For instance, $Window_{t-1}$ has anomaly score of S_{t-1} on samples at $x_{t-7}, x_{t-5}, x_{t-3}$ and x_{t-1}. The overall anomaly score of each point at this band is calculated as the sum of all windows that has anomaly score at that point.

distance between these two signals. Since the sliding is set to 1 so, for each sample of the test data, an anomaly score is generated with encoding error. Additionally, cosine similarity of the encoded signal and input data is calculated to increase the anomaly score performance (see Fig. 6). The cosine similarity calculation of two given vectors A and B is presented in (6):

$$cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\Sigma_{i=1}^{n} A_i B_i}{\sqrt{\Sigma_{i=1}^{n} A_i^2}\sqrt{\Sigma_{i=1}^{n} B_i^2}} \tag{6}$$

Cosine similarity results have a range of -1 to $+1$, where -1 means exactly opposite direction, $+1$ means exactly the same direction, and 0 means orthogonality. We utilize cosine similarity especially to observe small distorted anomalies. Encoding error between input and encoded signals can be substantially small at certain points depending of the characteristics of the data. In this case, if there is anomaly at some particular points, the anomaly can be detected by the cosine similarity approach, despite the fact that encoding error may be small. Here, to distinguish the signals that are not vectorially similar, the cosine similarity score for each band is subtracted from 1.05 and the square of the subtracted value is taken into consideration with (7), where S_c is the cosine similarity and cs is the cosine similarity score:

$$cs = (1.05 - S_c)^2 \tag{7}$$

We present an example in Fig. 7 that demonstrates how the cosine similarity helps to find anomalies, especially when the encoding error is small while the vectoral distances are high. We hereby aim to have a basic anomaly score calculation function to reduce the time complexity. Therefore, we simply multiply all cosine similarity scores, cs, and encoding errors, e, for each band to find anomaly score of that point. The complex AI techniques, such as ANN can also be used to determine the location of anomaly towards enhancing predictive characteristics of our methodology. However, here in this first step, we keep the approach simple, by simply multiplying each band's encoding errors and cosine scores, as presented in (8).

$$AS_i = \prod_{j=1}^{b} e_{i_j} \times cs_{i_j} \tag{8}$$

where, b is the band number, i is i$^{\text{th}}$ sample point of the test data set, and AS is the overall anomaly score. The window size is selected as 8 and the sliding parameter is set to 1 as discussed earlier, therefore, this results in overlapping multiple anomaly scores at each sample. To overcome this problem, we take the sum of overlapping windows at the same data sample, presented in Fig. 8. An example of anomaly detection scores for each band and final score in a sample data set is presented in detail in Fig. 9.

C. DATA SET

As we indicated earlier, we demonstrate the anomaly detection performance of the cortical coding network by using UCR Time-Series Anomaly Detection Benchmark [38]. This data set is introduced as a preference over the commonly used anomaly detection data sets which is free of the common data set flaws such as triviality, unrealistic anomaly density or mislabeled ground truth [36]. The UCR data set is carefully created to contain real data and includes different types of anomalies. It also contains noisy or distorted variations for the purpose of evaluating the performance of our method introduced herein under different circumstances. Since the data consists of 250 time-series, the framed window sizes of the input data sets vary from 103 to 249992 with mean 20986 and standard deviation 32418. Each data set is designed to have only one anomaly window and a

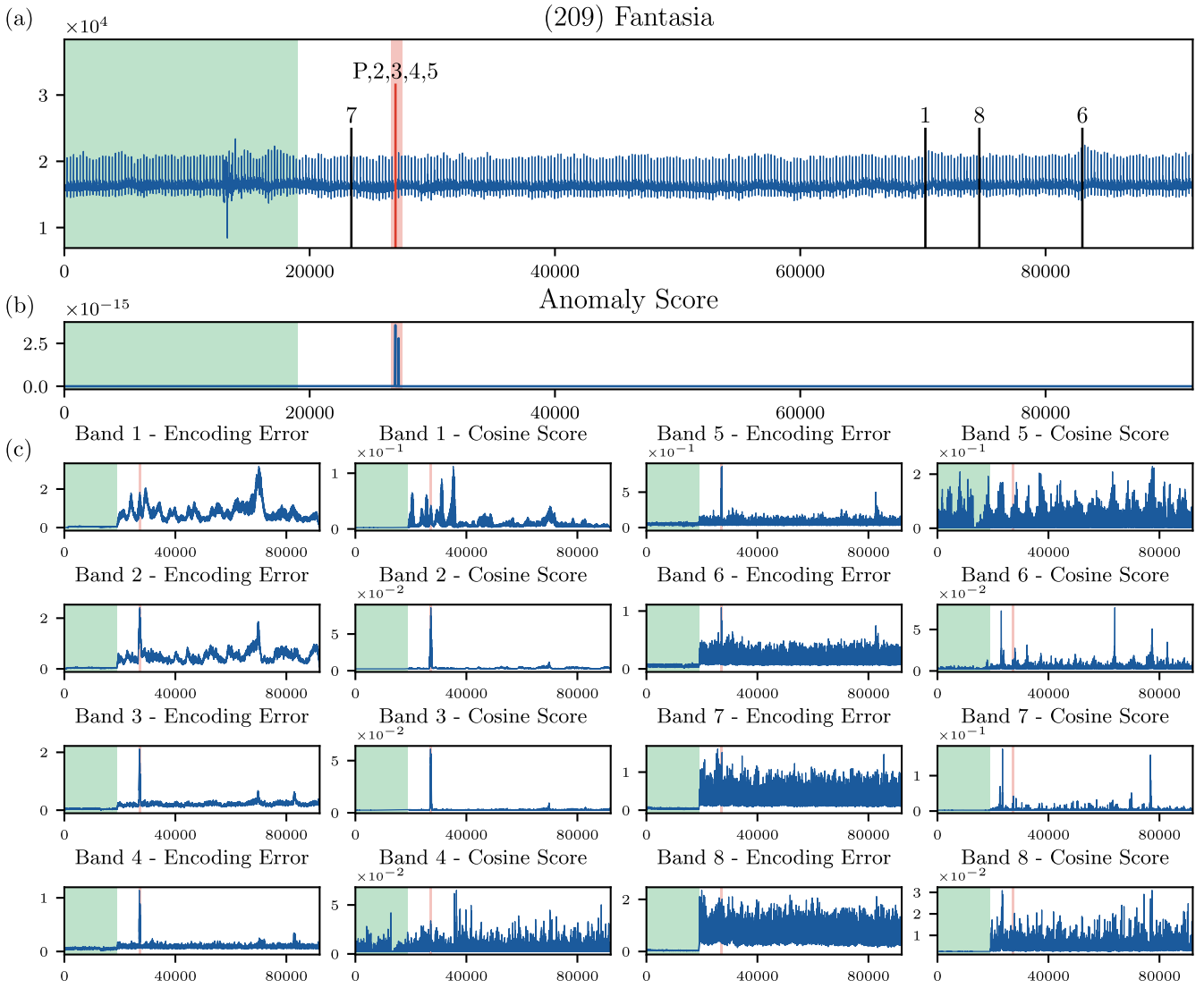


FIGURE 9. An example of anomaly score prediction in data set Fantasia. The green-highlighted area shows the training period of the data set, and pink is the anomaly window. (a) Input Data and anomaly score predictions from each band ($b = 1, 2, \ldots, 8$) and the overall anomaly prediction (the multiplication of each band scores) P are presented. The predictions from each band (black lines with number 1 to 8) shows the highest anomaly score point of relevant band ($\text{argmax}(e_{b_i} \times cs_{b_i})$). The product of each band P correctly predicts anomaly as well as 2nd, 3rd, 4th, and 5th bands. (b) Total anomaly score of each point. Highest predicted point is inside the pink area, therefore anomaly is correctly predicted. (c) The detailed view of each band's encoding error and cosine score. Here, the anomaly is truly detected for both e and cs on 2nd and 3rd bands while it was only truly detected for e on 4th, 5th, and 6th bands.

predetermined training period that is free of anomalies. We aim to find the data point with the maximum anomaly score in each data set. This data point is assumed to be a truly detected anomaly if it is within the detection range, represented in Fig. 10. The total score using the UCR anomaly benchmark data set is calculated as the percentage of the correct anomaly prediction count over all of the data sets.

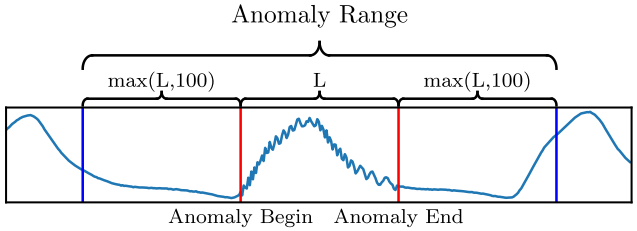


FIGURE 10. Anomaly detection range in the UCR anomaly benchmark data set.

D. METHODS USED FOR PERFORMANCE COMPARISON VS CORTICAL CODING

Various clustering algorithms are chosen to compare their performance versus the cortical coding approach. Two main criteria are set to choose the algorithms used for comparison. The first criterion for choice was to look for popular clustering algorithms from a variety of taxonomy of clusters.

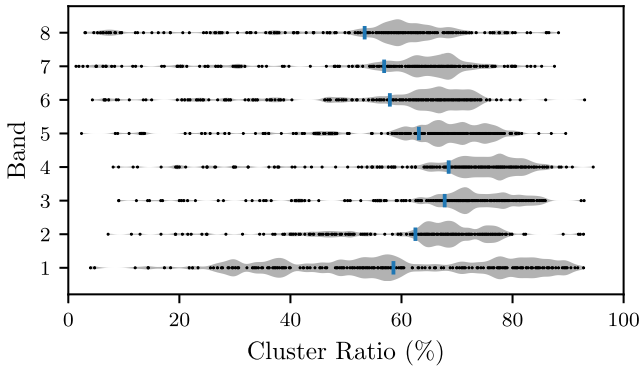


FIGURE 11. Clustering performance of each band for all the 250 data sets from UCR benchmark data set. Blue line indicates mean clustering ratio of the related band.

The second criterion was that the chosen clustering method would use the cluster size as an input parameter. This is a crucial factor, because one would then ensure a fair comparison of the method chosen which would have the same cluster sizes as that of the cortical coding method. As a result, from the partition-based clustering algorithms, we chose the most popular ones, i.e., K-means and its variant mini-batch K-means. Among the hierarchical clustering methods, we choose hierarchical agglomerative clustering as the representative of this class of algorithms. All the chosen methods work in offline manner, i.e., each needs to analyze all input data in advance. Since the cortical coding method works in truly online manner, another algorithm, i.e., sequential K-means is added to the methods used for comparison. The last method was chosen which would work without performing any clustering, i.e., first, all the input data is memorized and then the performance is analyzed to figure out a baseline comparison. In short, therefore, five different methods were used to assess the performance of the cortical coding method.

The input signals are divided into 8 frequency bands at the processing stage of the cortical coding method used in anomaly detection. Our method uses 8 frequency bands for each of the 250 benchmark data sets used resulting in a total of 2, 000 cluster points. The statistics of the clustering ratio of all bands of data sets are the minimum of 1.38% and the maximum of 94.53% with 61.08% mean and 18.38% standard deviation. A schematic showing the distribution of clustering performance is presented in Fig. 11. Below we describe each of the methods used for comparison in some detail.

1) K-MEANS

K-means is one of the oldest and most popular clustering algorithms due to its simplicity and ease of implementation [45]. It is widely used in data mining and clustering areas such as business intelligence, pattern classification, social science, anomaly detection, etc. [45], [46]. The K-means algorithm aims to partition n objects $(x_1, \ldots, x_n)$ into k clusters. The standard algorithms randomly assigns k cluster centroids c_i, where $i = 1, 2, \ldots, k$, and finds for each x_j the closest centroid point in Euclidean space ($argmin = \|\vec{x}_j - \vec{c}_i\|$). The mean of the each cluster centroid is recalculated with the nearest data points. The iterations of finding nearest cluster points and recalculating the means of the centroids steps continue until the convergence condition is met. The drawbacks of the algorithm include the dependence of its overall performance on the initialization of the centroids and the requirement of reaching a global optimum leading to an uncertain solution. Time and space complexity of the K-means is considerably good, time complexity is $O(nkdi)$ and space complexity is $O(n + k)d$, where n is the size of input, k is the number of clusters, d is the dimension of the vector, and i is the number of iteration. In practice, K-means is frequently run several times and the average of the results are taken into consideration to reduce the drawback due to random initialization problem. Alternatively, K-means++ can be used for the initialization of the centroids, as it is performed in this work. The pseudo code of the algorithm and the parameters used for the K-means tests are presented in Appendix A.

2) MINI-BATCH K-MEANS

Mini-batch K-means algorithm is a well-known variant of K-means, mainly used for clustering massive data sets. Even though time complexity of K-means is not so high, when the data amount increases, the computational cost of each iteration increases. Instead of calculating all distances in each iteration of K-means, a fixed size sub-sample of data is taken into consideration and used for iterations in mini-batch K-means algorithm. In this case, therefore, the required amount of calculations for each iteration are reduced while speeding up the overall algorithm but with a possible cost of low clustering performance [47], [48]. The pseudo code of the algorithm and the parameters used for mini-batch K-means tests are presented in Appendix A.

3) HIERARCHICAL AGGLOMERATIVE CLUSTERING

Hierarchical Agglomerative Clustering is the most popular Hierarchical Clustering method which has generally a good clustering performance. Although it is a deterministic model, it is also a greedy, high-energy consuming model with high time and space complexity [49]. At the initial step of the algorithm, each n observations of the input data treated as the clusters (singleton) and the most similar data points are merged at each iteration until termination condition is reached. The termination condition can be a condition at which there would be no cluster pair to be merged left, a desired number of clusters (k) reached or a threshold of the distance value. To find the most similar cluster pairs, cluster to cluster distances are calculated and these are stored in a distance matrix. Since the distances are symmetric, calculating only the bottom-up part of the matrix is enough for the analysis. At each iteration, newly generated merged cluster distance to other clusters are calculated and the

distance matrix is updated. There are various linkage methods utilized to determine which clusters to be merged, such as single linkage, complete linkage, Ward's linkage etc. [50]. Ward's linkage, also referred as minimum variance method, is used in this work, as it analyzes the variance of clusters instead of directly use the distance measures. The naive hierarchical agglomerative algorithm has time complexity of $O(n^3)$ which, with some improvements, can be reduced to $O(n^2 logn)$ [51]. The space complexity of Hierarchical Agglomerative Clustering algorithm is $O(n^2)$. However, this method is not suitable for clustering when the large data is used as the input, because, in these cases, the memory requirements and the run-time of the algorithm become prohibitively too large for practical implementation. The pseudo code of the algorithm and the parameters used in tests are presented in Appendix A.

4) SEQUENTIAL K-MEANS

Sequential K-means is the online working modification of K-means discussed above. Different from K-means, however, the input vectors are given to the algorithm one at a time rather than all at once. The method is used mostly in streaming clustering [52], [53]. Since the cortical coding method works online in sequential manner, like sequential K-means, its use is a fairer comparison than comparing offline clustering algorithms. Sequential K-means has time complexity of $O(n)$ and space complexity of $\Omega(kd)$ where n is the number of input, k is the number of clusters and d is the vector size. As with other methods, the pseudo code of the algorithm is presented in Appendix A.

5) MEMORIZING ALL OF THE INPUT DATA SET

All the input data sets are directly processed without performing any clustering in the final method used for performance comparison versus the cortical coding network. Although this method may not be considered as a fair comparison since the cluster sizes are not equal to those of the other algorithms used for comparison, it is presented to establish that clustering is performed not just to decrease the amount of the input data but rather to learn the characteristics of the data. Therefore, memorizing all input train data set will give the baseline outlook about the performance of cortical coding method used for anomaly detection in this paper.

III. RESULTS AND DISCUSSION

Anomaly detection, in general, has two main trade-offs with respect to classification problems in machine learning. The first is the presence of a wide, unlimited variety of possible anomaly patterns that are different than the set of limited, less frequently encountered cases, or states, that are accepted as ''normal'' or ''healthy.'' The second is the availability of data sets with relatively small known and labeled anomalous cases in comparison to overall past data sets. Zero-day exploits in cybersecurity may be shown as an example to these trade-offs. Higher performance anomaly detection requires the identification of the system rather than classification of known patterns at its outputs. In many cases, target system itself can be classified as time-varying and non-linear. For example, an accurate anomaly detection in predictive maintenance corresponds to finding the time varying system states caused by a developing failure. Another difficulty of the problem is the fact that there are large scales in time and the frequency domains. Anomalous behavior can either be a part of slow developing progress besides the active high output bandwidth or instantaneous variation in a large time window. Here, we have demonstrated a new approach inspired by the cortical networks in the brain for highly accurate and computationally efficient anomaly detection in potential real-time applications. Since the new method is based on self-structured cortical network involving cortex nodes, it also provides scalability in parallel processing. This important feature supports in-memory processing capability that is necessary for energy efficient implementation of neuromorphic processors besides the multi-core multi-CPU applications in conventional computational systems.

Hyperparameter-free solution presented here provides an example for the initial proof of the newly introduced concept demonstrating that the cortical coding is a promising method for finding anomalies with less dependency to subjective qualifications. The anomaly detection method developed has the potential of having even higher reliability than the average human sensitivity and intelligence.

An online Multi-data set Time Series Anomaly Detection competition was held by Hexagon-ml. More than 500 teams and 700 competitors took part in the competition and 193 competitors submitted their results [39]. The winner achieved the score of 88.4%, where the 10th received the score of 80%. If the work presented here had have joined the competition, our rank would had been 11th with the score of 77.6%. Final score using UCR data set is presented in Table 2. As given in (8), the final anomaly score is the product of each band's score. Using only the encoding error, our methodology scored 58.4%. When the anomaly scores of overlapping windows are also taken into consideration, the total score increases by 6%, resulting in the overall score of 64.4%. Finally, incorporating the cosine similarity in to the methodology enhanced the performance significantly, as shown by the increased overall score of 77.6%. We emphasize here that by considering the anomaly scores for each band on their own can drastically improve the final score even more. In the second and third sections of Table 2, we show the prediction accuracy for individual bands.

The results in the mid-section of the Table 2 indicate that the correct anomalies are found at least one band in 89.6% of the data set. This is important and may be considered as the theoretical maximum of the cortical coding method even without optimizing the preprocessing stage. The bottom section of the table shows the anomalies found from each band individually. The anomalies found in low frequency bands are less than the average of the other bands. This verifies that the frequency band divisions in the

TABLE 2. Performance results of cortical coding method.

	Correct Prediction	Accuracy (%)
Product of each band		
Only Encoding Error (e)	146	58.4
(e) and overlapping	161	64.4
e, cs and overlapping	194	77.6
For each band (*e*, *cs* and overlapping)		
At least one band	223	89.2
At least two bands	200	80
At least three bands	179	71.6
At least four bands	150	60
Individual bands		
Band 1	67	26.8
Band 2	103	41.2
Band 3	135	54.0
Band 4	133	53.2
Band 5	151	60.4
Band 6	127	50.8
Band 7	129	51.6
Band 8	124	49.6

processing stage can be organized better to optimize the frequency ranges to achieve maximum performance from each band. The statistical analyses of the training period for each data set may yield better frequency range optimization for filtering. Optimizing the frequency ranges, therefore, would be expected to further increase the overall score. To demonstrate this point, in Fig. 12, we give an example of the correct prediction of a wrongly predicted anomaly by arranging frequency band ranges in the preprocessing stage. Again we emphasize that, in the current work, the score of 77.6% has been achieved without such simple factors discussed above which would inevitably improve the final score even further.

The results of all the algorithms used for performance comparison are presented in Table 3. As seen from the table, the cortical coding method score is better than the all other methods compared where Hierarchical Agglomerative Clustering has the closest score (74%). K-means and variants of K-means could not exceed the score of 64%. Surprisingly, mini-batch K-means results are slightly better than K-means results which may be interpreted as that better clustering performance does not always reflect better anomaly score. In addition, another factor supporting this interpretation is due to the result that memorizing the data set (MEM) results found out to be worse than all standard offline clustering algorithms. Since there is no clustering performed in MEM method, it has the advantage of having larger codebook than the rest of the algorithms, yet it performs the worst among the offline clustering algorithms, having only slightly better score than sequential K-means. The significance of learning by clustering rather than memorizing, therefore, can be emphasized from these results. As expected, sequential K-means method scored worst due to the disadvantage of the online clustering, however, having the same disadvantage of online clustering, the cortical coding method surpasses those of the other methods used. The detailed frequency band analyses of the results are presented in Appendix B.

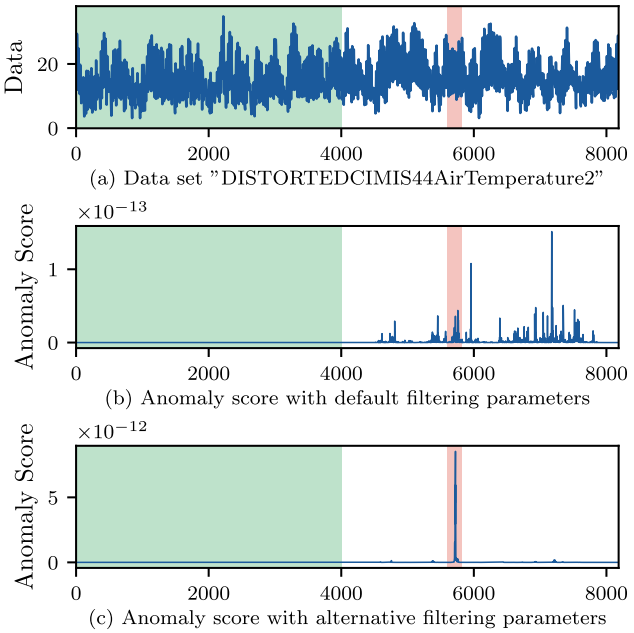


FIGURE 12. Frequency band arrangement in the training stage corrects the wrongly predicted anomaly. This process potentially improves the overall score of anomaly prediction. (a) The input data from the UCR data set, where the green window shows the training period and the anomaly window is highlighted with pink. (b) The default frequency band division used in this paper results in a wrong prediction for this data set. The highest anomaly score is selected as the predicted anomaly. (c) If an alternative frequency division process is used, which analyzes the dominant frequencies in the training period, then the correct anomaly can be found clearly for the data set. This process, when implemented, would increase the predictive score even further.

TABLE 3. Results of all methods.

	Correct Prediction	Accuracy (%)
Cortical Coding (COR)	194	77.6
K-means (KMN)	153	61.2
Mini-batch K-means (MBK)	160	64
Hierarchical Agglomerative (HAC)	185	74
Sequential K-means (SEQ)	148	59.2
Memorizing (MEM)	152	60.8

All the individual results for each the 250 data set is presented in Fig. 13 for a better understanding of the results from the UCR data, where, the data sets are sorted according to their data size from the smallest to the largest. It can be noticed that as the data size increases, the truly detected anomalies decrease. All the methods fail to find the correct anomaly in 39 out of 250 data sets where 142 of them are correctly predicted by all methods. The same table with the actual data set number from UCR Anomaly Benchmark Data set is presented in Appendix B without performing a sort option.

The overarching goal of this study has not been to have best score using UCR data set, but rather to present our brain inspired cortical coding method in anomaly detection and provide potential future opportunities using it in its most basic form. Due to the advantages of our method having low time

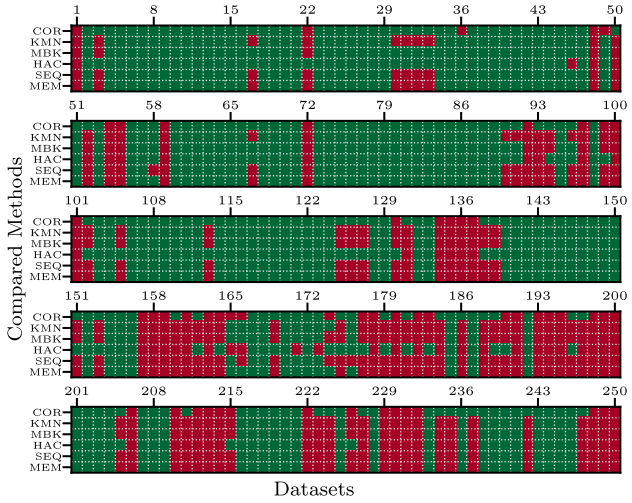


FIGURE 13. Detailed performance comparison of algorithms for each data set sorted by data size from the smallest to the largest. Green square indicates correct anomaly prediction while red indicates wrongly detected anomaly.

complexity and being highly suitable for online clustering, we emphasize here that it could be easily implemented in unsupervised anomaly detection problems. Even though the performance of the cortical coding method is the highest among all the methods used, the study presented herein does not reflect the best achievable values in its current form since several advantages have not been considered. For example, one may consider introducing hyperparameter tuning, especially in the preprocessing stage. To demonstrate the ways to improve the overall anomaly detection accuracy of the cortical coding method even further, below we provide two main suggestions:

- The use of data set specific filtering: This depends on the characteristics of each data set instead of utilizing the predetermined frequency ranges in filtering stage. We filter all of the 250 data sets used in this study with the same frequency ranges. In the training stage, analyzing the distribution of the frequencies for each data set and arranging filtering bands according to the analysis would most probably increase the overall accuracy. For instance, the distribution of the frequencies in some data set are significantly narrow. Dividing some of the frequency bands with wider ranges would result in non-functional frequency bands which are taken into consideration to calculate the final anomaly score. In Fig. 12, we provide an example of a wrongly predicted data set that may be turned into the correct prediction by introducing specialized frequency bands.
- The use of more sophisticated anomaly score calculation: In this study, we multiply all the anomaly scores for each band with each other to achieve the final anomaly score since we aim to keep the anomaly score function simple. Here, the weight of the score for each band is kept the same. Determining the dominant frequencies and by arranging each frequency band with the varying weighted scores may yield better results. Moreover, one can also use a machine learning algorithm, e.g., Multi Layer Perceptron (MLP), to calculate the anomaly score as well. Alternatively, one may use a combination of our anomaly score calculation function with a machine learning method.

IV. CONCLUSION AND FUTURE WORK

In this study, a brain inspired cortical coding method is introduced by demonstrating its high performance in time series anomaly detection despite its low time and space complexity. The evaluation was accomplished using a comprehensive benchmark data set covering a wide range of frequently encountered anomalies. The overarching goal here has been to present the generalization performance of the developed method in various time series anomaly detection problems by comparing it with frequently used algorithms. In the first step, we used the same initial hyperparameter set for all 250 problems in the benchmark data set; this approach is different from the problem-specific tuning or dedicated solutions, led to 77.6% performance, top score among all compared methods, including K-means (61.2%), mini-batch K-means (64%), Hierarchical Agglomerative Clustering (74%), Sequential K-means (59.2%) and memorizing data (60.8%). Additionally, based on the on-line competition [39], the non-problem-specific solution performance rate of the new method is within the first 5 percentile of the competing other methods. More significantly, despite having best score among the compared methods, the time and space complexity of the cortical coding method is one of the best. The demonstrated generalization capability of the proposed method appears to be a promising alternative for non-problem-specific solution in time series anomaly detection problems. On the other hand, there is still a significant room for possible improvements regarding to problem oriented hyperparameter set adaptations. We herein presented some examples on how such enhancements would improve the results.

We emphasize that the primary advantage of the cortical coding network method is its low time and space complexity besides the good generalization performance. Additionally, there are three other advantages in the cortical coding process. The first is that it has the compatibility to the in-memory processing; this capability would provide new opportunities especially facilitated by neuromorphic computing. The second advantage is the structuring of the network. This is related to the deep feature relationships and their probabilistic distributions depending on the interaction with the physical environment. This characteristics facilitates an opportunity to avoid computational load of unnecessary weights in the matrix type operations, as in the case of conventional machine learning methods. Thirdly, the method works in online manner, which makes it suitable for stream clustering, and real-time applications. In consequence, the cortical coding network model enables high computational efficiency

with scalable and easy-to-implement parallel computational structures.

The fundamental architecture of the cortical coding model facilitates numerous further opportunities for additional improvements with more efficient and fast information processing. In the next step, for example, the anomaly detection efficiency may be enhanced by assigning the frequency range for each band individually (see Fig. 12). The current study is the first level demonstration of the cortical coding network, functioning as the feature extractor accompanied with the encoding error evaluator for anomaly detection. We plan to add additional layers to the network to gain the capability to solve more complex multi-variate time series anomaly detection problems and reasoning, e.g., towards more context-based results in the future such as used in predictive maintenance, cybersecurity, telemedicine and risk management.

APPENDIX A
CLUSTERING METHODS USED IN THE MODEL

The pseudo codes and the parameters of the algorithms used for comparison are explained in this chapter, the source codes are available in Github repository [54]. The pseudo code of the Cortical Coding Method for one frame is presented in Algorithm 1. The default parameters of K-means in scikit-learn library are used for K-means tests [55]. As discussed earlier in II-D in Methodology, one of the major drawbacks of the standard K-means algorithm is the random initialization of cluster centroids. There are various efficient initialization methods for K-means developed to overcome this drawback. One of the most successful and popular one is K-means++. Instead of randomly choosing all k cluster centers with equal probability, in K-means++ only the first cluster center is randomly chosen and the rest $k-1$ cluster centers are chosen with a probability function $\frac{D(x')^2}{\sum_{i=1}^{n} D(x_i)}$ where n is the number of input data, k is the cluster size and $D(x)$ denotes the minimum distance between the point x and the closest cluster center. K-means++ initialization reduces the probability of choosing initial cluster centroids very close to each other, therefore it is likely to speed up the convergence [56], [57]. The pseudo code of the K-means with K-means++ initialization is presented in Algorithm 2.

Algorithm 1 Cortical Coding Network Training (for One Frame)

```
Input
    W   input wavelet coefficients vectors, {w_1, w_2, ..., w_d}
procedure Cortical Coding Network (Cortex) Training
    for each coefficent (w_i) in the input data frame do
        if node has progeny cortex node then
            closest_node ← FindClosestNode(node, coefficent)
            if coefficient is in the range of the closest cortex node then
                node ← UpdateClosestNode(closest_node, coefficient)
                continue
            end if
        end if
        if node has spines then
            closest_spine ← FindClosestSpine(node, coefficent)
            if coefficient is in the range of the closest spine node then
                UpdateClosestSpine(closest_spine, coefficent)
                if closest_spine's maturity > threshold then
                    node ← EvolveSpineToNode(closest_spine)
                    continue
                else
                    break
                end if
            end if
        end if
        GenerateSpine(node, coefficient)
        break
    end for
end procedure
```

Algorithm 2 K-Means

```
Input
    X   input vectors, {x_1, x_2, ..., x_n}
    k   number of clusters
Output
    C   centroids, {c_1, c_2, ..., c_k}
procedure k-means
    Initialize centroids ({c_1, c_2, ..., c_k}) with k-means++
    while not converged do
        Assign each vector in X to the closest centroid c
        for each c ∈ C do
            c ← mean of vectors assigned to c
        end for
    end while
end procedure
```

Mini-batch K-means is a variant of K-means, specifically used to cluster massive amounts of data. The default parameters from scikit-learn library are used for mini-batch K-means as well except batch size parameter. The default batch size is normally 1024. However, since the input data size is large for some of the UCR anomaly benchmark data set, the batch size parameter is set to 2048. The pseudo code of the mini-batch K-means algorithm is presented in Algorithm 3.

Algorithm 3 Mini-Batch K-Means

```
Input
    X   input vectors, {x_1, x_2, ..., x_n}
    k   number of clusters
    b   batch size
    t   number of iterations
Output
    C   centroids, {c_1, c_2, ..., c_k}
procedure mini-batch K-means
    Initialize centroids ({c_1, c_2, ..., c_k}) with k-means++
    for i ← 1...t do
        Pick random b input from X and assign to M
        Assign each vector in M to the closest centroid c
        for each c ∈ C do
            c ← mean of vectors assigned to c
        end for
    end for
end procedure
```

Hierarchical Agglomerative Clustering treats each data points as a cluster center at initial step and recursively merges cluster as a result of a similarity function until termination condition is reached. The default parameters of scikit-learn library is used for the tests [55]. Ward's linkage (minimizing the variance of the merged clusters) is used to determine the distance to be used for the similarity. The algorithm is run by applying number of clusters as a parameter. The pseudo code of the algorithm is presented in Algorithm 4.

Algorithm 4 Hierarchical Agglomerative Clustering (HAC)

Input
X input vectors, $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$
k number of clusters
Output
C centroids, $\{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k\}$
procedure hac
Assign each input as a cluster centroid ($\vec{c}_i \leftarrow \vec{x}_i$)
Compute proximity matrix by calculating the distance of each centroids
repeat
Merge the pair of cluster that has smallest distance
Update proximity matrix
until Termination condition is reached.
end procedure

The Sequential K-means algorithm is the online working variant of K-means algorithm where the input data points are accumulated progressively [58]. Sequential K-means can be implemented by using mini-batch K-means with setting batch size to 1 and using *partial_fit()* function from scikit-learn library. In this case, unlike standard Sequential K-means method presented in Algorithm 5, K-means++ is used for initialization. As discussed earlier, instead of randomly initialize (or picking first k inputs as initial centroids), using K-means++ is expected to result a better clustering performance. Therefore, K-means++ initializer is used for Sequential K-means tests.

Algorithm 5 Sequential K-Means

Input
X input vectors, $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$
k number of clusters
Output
C centroids, $\{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k\}$
procedure sequential k-means
Initialize centroids ($\{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k\}$) with first k inputs
for $i \leftarrow k \ldots n$ **do** ▷ while data is present
Assign next vector in x_i to the closest centroid c
Update assigned cluster centroid (mean of all assigned vectors)
end for
end procedure

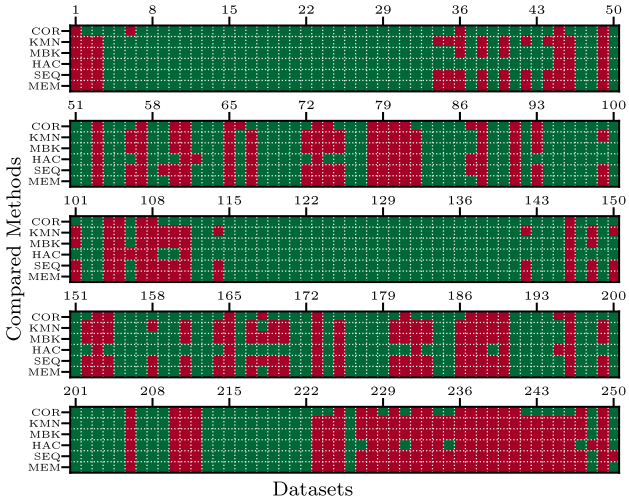


FIGURE 14. Detailed performance comparison of algorithms for each data set. Green square indicates correct anomaly prediction while red indicates wrongly detected anomaly.

TABLE 4. Detailed results of all methods used for comparison vs the cortical coding method.

	COR	KMN	MBK	HAC	SEQ	MEM
Overall Result (%)	77.6	61.2	64	74	59.2	60.8
For each band (%)						
At least 1 band	89.2	76.4	78	87.2	76	76
At least 2 bands	80	69.2	71.6	78.4	67.6	68.8
At least 3 bands	71.6	59.6	60	66.4	58	59.2
At least 4 bands	60	49.2	47.6	56.4	48	48.8
Individual bands (%)						
Band 1	26.8	14	16	20.8	13.6	14
Band 2	41.2	37.2	34.8	34.4	36.8	37.6
Band 3	54	42.8	42	41.2	43.2	42.8
Band 4	53.2	51.6	49.6	53.6	50.8	51.6
Band 5	60.4	52.4	54.4	58.8	51.2	52
Band 6	50.8	48	48.4	60.8	46.8	47.6
Band 7	51.6	47.2	48.8	57.6	46	46.8
Band 8	49.6	44	49.2	56.4	44.4	44

APPENDIX B
DETAILED RESULTS

Here we give Fig. 14 which is the non-sorted version of Fig. 13, the results are the same but the order of data set is different. We present the results with the actual data set number for the interested readers working with UCR data to easily compare by using data set numbers shown here.

We also show Table 4 which is the extended version of the performance results of all the 5 methods compared in this work. As discussed in Methodology, the total anomaly score is the product of each band's anomaly score, which is the function of encoding error and cosine similarity. With the extended results, we aim to present the individual band scores of all the methods. From the results, it can be interpreted that if the anomaly is found at least in 3 bands correctly, it is enough to predict the correct anomaly (All *At least 3 bands* results are smaller than *Overall Results*). One could note that by comparing individual band results, most successive bands appear to be mid-frequency bands. More specifically, band 5 has the highest score for all methods

compared except Hierarchical Agglomerative Clustering and it is runner-up score. More significantly, the high frequency bands' scores are better than low frequency results. This conclusion may reflect the characteristics of anomalies in UCR data set. Based on this view, a simple frequency division introduced accordingly (giving more importance to high frequency bands) would most probably increases the overall success rates of all methods.

ACKNOWLEDGMENT

The authors would like to thank Professor Mehmet Sarikaya for the inception, development, and revision of this manuscript and Selim Eren Eryilmaz for his technical assistance.

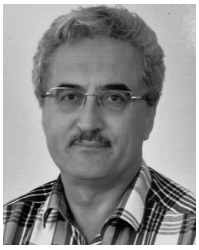
REFERENCES

[1] J. H. Hong, C.-C. Liu, and M. Govindarasu, "Integrated anomaly detection for cyber security of the substations," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 1643–1653, Jul. 2014.
[2] H. Karimipour, S. Geris, A. Dehghantanha, and H. Leung, "Intelligent anomaly detection for large-scale smart grids," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.
[3] V. Jyothsna, V. R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *Int. J. Comput. Appl.*, vol. 28, no. 7, pp. 26–35, Aug. 2011.
[4] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, nos. 1–2, pp. 18–28, 2009.
[5] J. V. Spiric, M. B. Docic, and S. S. Stankovic, "Fraud detection in registered electricity time series," *Int. J. Electr. Power Energy Syst.*, vol. 71, pp. 42–50, Oct. 2015.
[6] C. Han, L. Rundo, K. Murao, T. Noguchi, Y. Shimahara, Z. Á. Milacski, S. Koshino, E. Sala, H. Nakayama, and S. Satoh, "MADGAN: Unsupervised medical anomaly detection GAN using multiple adjacent brain MRI slice reconstruction," *BMC Bioinf.*, vol. 22, no. 2, pp. 1–20, Apr. 2021.
[7] P. J. Rousseeuw and M. Hubert, "Anomaly detection by robust statistics," *WIREs Data Mining Knowl. Discovery*, vol. 8, no. 2, p. e1236, Mar. 2018.
[8] M. Ahmed, A. N. Mahmood, and J. Hu, "Outlier detection," in *The State of the Art in Intrusion Prevention and Detection*. Boca Raton, FL, USA: CRC Press, 2014, pp. 3–21.
[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.
[10] C. M. Ahmed, M. R. G. Raman, and A. P. Mathur, "Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems," in *Proc. 6th ACM Cyber-Phys. Syst. Secur. Workshop*, Oct. 2020, pp. 23–29.
[11] M. Yucel, S. Bagis, A. Sertbas, M. Sarikaya, and B. Ustundag, "Brain inspired cortical coding method for fast clustering and codebook generation," *Entropy*, vol. 24, no. 11, p. 1678, Nov. 2022.
[12] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Proc. Comput. Sci.*, vol. 60, pp. 708–713, Jan. 2015.
[13] E. Keogh, S. Lonardi, and B. Y. C. Chiu, "Finding surprising patterns in a time series database in linear time and space," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2002, pp. 550–556.
[14] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Comput. Surveys*, vol. 54, no. 3, pp. 1–33, Apr. 2022.
[15] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Process.*, vol. 99, pp. 215–249, Jun. 2014.
[16] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, 2004.
[17] B. Rossi, S. Chren, B. Buhnova, and T. Pitner, "Anomaly detection in smart grid data: An experience report," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 002313–002318.
[18] M. A. Hayes and M. A. Capretz, "Contextual anomaly detection framework for big sensor data," *J. Big Data*, vol. 2, no. 1, pp. 1–22, Dec. 2015.
[19] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art," 2020, *arXiv:2004.00433*.
[20] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016.
[21] K. M. Biradar, A. Gupta, M. Mandal, and S. K. Vipparthi, "Challenges in time-stamp aware anomaly detection in traffic videos," 2019, *arXiv:1906.04574*.
[22] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proc. Internet Meas. Conf.*, 2015, pp. 211–224.
[23] G. Sebestyen, A. Hangan, Z. Czako, and G. Kovacs, "A taxonomy and platform for anomaly detection," in *Proc. IEEE Int. Conf. Autom., Quality Test., Robot. (AQTR)*, May 2018, pp. 1–6.
[24] M. Gyanchandani, J. L. Rana, and R. N. Yadav, "Taxonomy of anomaly based intrusion detection system: A review," *Int. J. Sci. Res.*, vol. 2, no. 12, pp. 1–13, Dec. 2012.
[25] H.-S. Wu, "A survey of research on anomaly detection for time series," in *Proc. 13th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. (ICCWAMTIP)*, Dec. 2016, pp. 426–431.
[26] K. Shaukat, T. M. Alam, S. Luo, S. Shabbir, I. A. Hameed, J. Li, S. K. Abbas, and U. Javed, "A review of time-series anomaly detection techniques: A step to future perspectives," in *Proc. Future Inf. Commun. Conf.* Cham, Switzerland: Springer, 2021, pp. 865–877.
[27] S. Lee and H. K. Kim, "ADSaS: Comprehensive real-time anomaly detection system," in *Proc. Int. Workshop Inf. Secur. Appl.* Cham, Switzerland: Springer, 2018, pp. 29–41.
[28] S. Omar, A. Ngadi, and H. H. Jebur, "Machine learning techniques for anomaly detection: An overview," *Int. J. Comput. Appl.*, vol. 79, no. 2, pp. 1–9, 2013.
[29] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine learning for anomaly detection: A systematic review," *IEEE Access*, vol. 9, pp. 78658–78700, 2021.
[30] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin, "Incremental anomaly detection using two-layer cluster-based structure," *Inf. Sci.*, vol. 429, pp. 315–331, Mar. 2018.
[31] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 22, pp. 949–961, Jan. 2019.
[32] M. Kim, J. Yun, Y. Cho, K. Shin, R. Jang, H.-J. Bae, and N. Kim, "Deep learning in medical imaging," *Neurospine*, vol. 16, no. 4, p. 657, 2019.
[33] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*.
[34] G. Pang, C. Shen, L. Cao, and A. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2020.
[35] G. Pang, L. Cao, and C. Aggarwal, "Deep learning for anomaly detection: Challenges, methods, and opportunities," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, Mar. 2021, pp. 1127–1130.
[36] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 14, 2021, doi: 10.1109/TKDE.2021.3112126.
[37] M. Yucel, S. Bagis, A. Sertbas, M. Sarikaya, and B. B. Ustundag, "Bioinspired cortex-based fast codebook generation," 2022, *arXiv:2201.12322*.
[38] E. Keogh. (Jun. 2021). *Time-Series Anomaly Detection Datasets*. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip
[39] E. Keogh, T. R. Dutta, U. Naik, and A. Agrawal. (2021). *Multi-Dataset Time-Series Anomaly Detection Competition, SIGKDD*. [Online]. Available: https://compete.hexagon-ml.com/practice/competition/39/
[40] R. M. Gray, *Entropy and Information Theory*. Berlin, Germany: Springer, 2011.
[41] J. C. Magee, "Dendritic integration of excitatory synaptic input," *Nature Rev. Neurosci.*, vol. 1, no. 3, pp. 181–190, Dec. 2000.
[42] J. C. Magee and C. Grienberger, "Synaptic plasticity forms and functions," *Annu. Rev. Neurosci.*, vol. 43, no. 1, pp. 95–117, Jul. 2020.
[43] S. Butterworth, "On the theory of filter amplifiers," *Wireless Engineer*, vol. 7, pp. 536–541, Oct. 1930.
[44] C. E. Heil and D. F. Walnut, "Continuous and discrete wavelet transforms," *SIAM Rev.*, vol. 31, no. 4, pp. 628–666, 1989.
[45] J. Wu, "Cluster analysis and K-means clustering: An introduction," in *Advances in K-Means Clustering: A Data Mining Thinking*. Berlin, Germany: Springer, 2012, pp. 1–16.
[46] M. Ahmed, R. Seraj, and S. M. S. Islam, "The K-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, Aug. 2020.

[47] A. Feizollah, N. B. Anuar, R. Salleh, and F. Amalina, "Comparative study of K-means and mini batch K-means clustering algorithms in Android malware detection using network traffic analysis," in *Proc. Int. Symp. Biometrics Secur. Technol. (ISBAST)*, Aug. 2014, pp. 193–197.
[48] D. Sculley, "Web-scale K-means clustering," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 1177–1178.
[49] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley Interdiscipl. Rev. Data Mining Knowl. Discovery*, vol. 2, no. 1, pp. 86–97, 2012.
[50] O. Yim and K. T. Ramdeen, "Hierarchical cluster analysis: Comparison of three linkage measures and application to psychological data," *Quant. Methods Psychol.*, vol. 11, no. 1, pp. 8–21, Feb. 2015.
[51] F. Nielsen, "Hierarchical clustering," in *Introduction to HPC With MPI for Data Science*. Berlin, Germany: Springer, 2016, pp. 195–211.
[52] A. F. Tehrani and D. Ahrens, "Modified sequential K-means clustering by utilizing response: A case study for fashion products," *Exp. Syst.*, vol. 34, no. 6, Dec. 2017, Art. no. e12226.
[53] A. King, "Online K-means clustering of nonstationary data," Course Project Rep., 2012. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7152498/
[54] M. Yucel. (Dec. 2022). *Time Series Anomaly Detection Using UCR Data Set*. [Online]. Available: https://github.com/mericy/AnomalyDetection
[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 10, pp. 2825–2830, Jul. 2017.
[56] S. Vassilvitskii and D. Arthur, "K-Means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2006, pp. 1027–1035.
[57] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the K-means clustering algorithm," *Exp. Syst. Appl.*, vol. 40, no. 1, pp. 200–210, Jan. 2013.
[58] E. Pomponi and A. Vinogradov, "A real-time approach to acoustic emission clustering," *Mech. Syst. Signal Process.*, vol. 40, no. 2, pp. 791–804, Nov. 2013.



MERIC YUCEL received the B.Sc. degree in computer engineering and the M.Sc. degree from Defence Technologies Program, Istanbul Technical University, Istanbul, Turkey, in 2005 and 2009, respectively. He is currently pursuing the Ph.D. degree with Istanbul University-Cerrahpasa under the supervision of Prof. Dr A. Sertbas and co-advisor B. B. Ustundag. He has been working as a Lecturer with the National Software Certification Research Center, Istanbul Technical University, since 2008. His research interests include signal processing, data science, machine learning, and neuromorphic computing.



AHMET SERTBAS was born in İstanbul, Turkey. He received the B.Sc. and M.Sc. degrees in electronic engineering from Istanbul Technical University, İstanbul, in 1986 and 1990, respectively, and the Ph.D. degree in electric-electronic engineering from İstanbul University, İstanbul, in 1997. Since 2000, he has been an Assistant Professor, an Associate Professor, and a Professor with the Computer Engineering Department, İstanbul University, and a Professor with the Computer Engineering Department, İstanbul University-Cerrahpasa, since 2018. His research interests include image processing, artificial intelligence, computer arithmetic, and hardware security. He has 22 articles in indexed SCI-SCIE journals and many journal articles not indexed SCI-SCIE and international conference papers.



BURAK BERK USTUNDAG (Member, IEEE) received the B.Sc. degree in electrical engineering and the M.Sc. and Ph.D. degrees in control systems and computer engineering from Istanbul Technical University (ITU). He is currently a Professor with the Computer Engineering Department, Faculty of Computer and Informatics, ITU. He has served as a Science and Technology Advisor to Ministers and Governmental Institutions for more than 15 years. He has more than 100 scientific publications and patents. His research interests include data fusion, artificial intelligence, signal processing, global optimization, cognitive communication, and agricultural information systems. He is a member of the IEEE Communication Society and Computational Intelligence.