IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 January 2023, accepted 13 February 2023, date of publication 22 February 2023, date of current version 28 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3247564

RESEARCH ARTICLE

Autoencoder-Based Iterative Modeling and Multivariate Time-Series Subsequence Clustering Algorithm

JONAS KÖHNE[1,2], LARS HENNING[2], AND CLEMENS GÜHMANN[1]
[1]Chair of Electronic Measurement and Diagnostic Technology, Technische Universität Berlin, 10623 Berlin, Germany
[2]IAV GmbH, 10587 Berlin, Germany

Corresponding author: Jonas Köhne (j.koehne@tu-berlin.de)

ABSTRACT This paper introduces an algorithm for the detection of change-points and the identification of the corresponding subsequences in transient multivariate time-series data (MTSD). The analysis of such data has become increasingly important due to growing availability in many industrial fields. Labeling, sorting or filtering highly transient measurement data for training Condition-based Maintenance (CbM) models is cumbersome and error-prone. For some applications it can be sufficient to filter measurements by simple thresholds or finding change-points based on changes in mean value and variation. But a robust diagnosis of a component within a component group for example, which has a complex non-linear correlation between multiple sensor values, a simple approach would not be feasible. No meaningful and coherent measurement data, which could be used for training a CbM model, would emerge. Therefore, we introduce an algorithm that uses a recurrent neural network (RNN) based Autoencoder (AE) which is iteratively trained on incoming data. The scoring function uses the reconstruction error and latent space information. A model of the identified subsequence is saved and used for recognition of repeating subsequences as well as fast offline clustering. For evaluation, we propose a new similarity measure based on the curvature for a more intuitive time-series subsequence clustering metric. A comparison with seven other state-of-the-art algorithms and eight datasets shows the capability and the increased performance of our algorithm to cluster MTSD online and offline in conjunction with mechatronic systems.

INDEX TERMS Condition-based maintenance, multivariate time-series data, change point detection, unsupervised clustering, autoencoder, segmentation, subsequence, clustering.

I. INTRODUCTION

In the applications of machine diagnosis of mechatronic systems and the subfield CbM, all supervised machine learning methods rely on high-quality labeled data [1], [2]. An option for a mechatronic system with different operating points is to measure the main operating points separately and create a diagnosis method for each of those individually. This requires a well-structured design and execution of experiments with a measurement labeling process. In a real world development environment for mechatronic system, where measurements are taken either automatically and/or manually and by many individuals and in different hardware and software development stages, providing consistently labeled and categorized data is a challenge.

Automatically labeling and categorizing multivariate time-series (MTS) data is therefore not only an alleviation but might be crucial for a successful CbM approach. As described above, labeled and categorized data is essential for training a model to represent a mechatronic system in a data driven approach. In the automotive sector where a lot of measurements occur at different operating points this is especially important. Some of these measurements are being recorded on a test bench in standard environment conditions with predefined operating points and for a given time (e.g., Worldwide harmonized Light Duty Test Cycle (WLTC)). Others can be

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

in idle mode during waiting or preparation time for a longer trip or other measurements. Also, very transient episodes are existent (e.g., Real Driving Emissions (RDE)). All of these measurements do not necessarily have the same calibration of the underlying mechatronic system. To train a robust model of the mechatronic system, component group or a single component, a big effort has to be put in the design of the experiments alone, not to mention the experiments themselves. Therefore, a method of automatically labeling existing measurements is of advantage. Afterwards an automatic sorting of the labeled time sequences by statistical methods is possible, to enable a data driven mechatronic diagnosis approach.

Using advanced unsupervised approaches for CbM allows the data to be unlabeled (otherwise supervised methods could be used). In this case the labeling refers to the label of the *condition* (mechanical degradation) of the monitored system. When trying to diagnose mechatronic systems that have many operating points and are free to transfer in between those or are capable of totally transient operation modes, then a robust diagnosis of the actual *condition* of the mechatronic system is extremely challenging. An early and reliable (robust) diagnosis of a mechatronic system prevents accidents, enables optimal maintenance and increases uptime of machinery. Without the knowledge of the current *condition* of the system, fault prevention can only be done by predetermined maintenance intervals. Motivation is therefore to monitor the health *condition* of the mechatronic system as close as possible, resulting in the task of separating discrete sensory data into uniquely identifiable and recognizable segments or subsequences. This is beneficial to the performance of *anomaly detection*, because if all normally occurring subsequences are identified, the detection of abnormal or faulty subsequences is straightforward.

When monitoring the health condition of a mechatronic system it is state of the art, to manually calibrate specific release conditions, during which the condition monitoring is enabled. This is done to exclude operating points which are very rare, too transient or are just not feasible for drawing conclusions about the *condition* of the mechatronic system. But even restricting the conditions on where to diagnose the machine (which already is reducing the probability of diagnosing the machine at all due to an operating state which is by chance outside the release conditions) cannot always help to improve the fault detection, identification and quantification of its magnitude. For example, in a mechatronic system with a complex nonlinear dependency of its subcomponents and its time dependency, ''going in'' or ''going out'' of the release conditions can result in very different system behavior. Comparing these two states does not lead to reliable conclusions for the mechatronic systems health *condition*.

Therefore, the kind of data sequences used for training/calibration and validation is crucial for any monitoring strategy. Manually screening, labeling and sorting data into comparable sequences is time-consuming, error-prone and cumbersome. Additionally, this is a decision process which requires expert and domain knowledge.

The new algorithm which we introduce in this work is capable of generating subsequence models from online streaming data which is processed sequentially. Any coherent subsequence that is identified can be recognized (clustered) if occurring again. Depending on multiple *sensitivity* calibration parameters, time-varying data points are associated and identified as a subsequence. The parameters determine the *volatility* or the strength of the affiliation required to be recognized as one time varying subsequence. These subsequence models can also be applied efficiently offline onto large existing datasets. During this prediction phase, the algorithm provides a vector of subsequence labels which were recognized as one from the training data. Depending on the calibration, it can also provide a label for *unknown* data which represents a phase where no pattern could be recognized. Otherwise, it finds the best fitting subsequence and labels it as that. The approach published in this work is currently only based on MTS input but could be adapted for a univariate input. It is a multivariate time-series sub-sequence discovery and identification method.

Our contribution is a new algorithm for online subsequence clustering of MTSD called ''Autoencoder-based Iterative Modeling and Subsequence Clustering Algorithm (ABIMCA)'' and a new metric to evaluate cluster algorithms focused on this task ''Multivariate Time-Series Sub-Sequence Clustering Metric (*MT3SCM*)''. We compare our algorithm with

- seven other state-of-the-art algorithms
- eight datasets, from which six are publicly available and two are provided with our codebase
- three widely used unsupervised clustering metrics
- our own metric (*MT3SCM*) and its four components

while varying the use of default algorithm parameters with optimized parameters on each algorithm and dataset via random grid search.

II. RELATED WORK

In this section we define the terminology used and its semantics to categorize our work within the large bibliography existing in this field and provide a selected list of related works and their ascendancy to this paper.

A. TERMINOLOGY AND SEMANTICS

Numerous possibilities have been described for achieving our main goal of segmenting discrete time-series sensory data. Most approaches can be sorted into the following partially overlapping categories: *time-series analysis* [3], *pattern recognition* [4], *temporal knowledge discovery* [5], *motif discovery* [6], *change-point detection* [7], *data clustering* [8] or *anomaly detection* [9]. All those terms refer to methods or algorithms which could be used directly or indirectly to achieve our goal. Explicit description of each term or category can be found in the stated references.

To limit the scope of this work, we focus on *data clustering* which can be separated into six subcategories by the

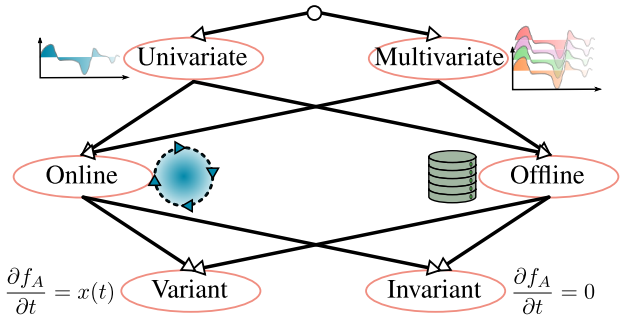

FIGURE 1. Combination possibilities of time-series clustering categories.

following groups of two: *univariate* and *multivariate data*, *online* and *offline* algorithms, *variant* and *invariant* data. The term "clustering" implies an unsupervised method. The equivalent supervised method would be called "classification". In the relevant literature more and other distinctions are made, depending on the specific field and context. First, we describe time-series data (TSD) since this is the data format we focus on in this paper. Afterwards we explain the differences between the subcategories.

"A time-series is a sequence of observations taken sequentially in time." [3] We denote a time-series data point as an observation with exactly one connected timestamp. The timestamp is not a variable or feature.

1) UNIVARIATE – MULTIVARIATE

If a single value or a scalar is the only variable of the data, then the data is univariate. It is the most basic format data can have. Considering TSD, a single temperature sensor with a timestamp would be univariate. Univariate TSD could also be interpreted as multivariate data of two dimensions, when taking the timestamp as another variable or feature.

2) ONLINE – OFFLINE

The differentiation between online and offline algorithms or analysis of data is crucial. Offline refers to data analysis that is applied to all the data at once. Measurement data, for example, is available in one or multiple files or can be accessed via a previously filled database. Offline algorithms can therefore iterate and optimize their result based on a criterion applied to known data. Online analysis on the other hand, is applied sequentially. The algorithm needs to be able to function with a criterion that generalizes well with unknown data. One selection or piece of data can be applied on the online algorithm without knowing the rest of the data. This approach cannot be as robust and accurate as an offline analysis, which is why most methods found in the literature are offline algorithms. Ideally the online algorithm learns as new data is provided. For the sake of completeness, however, it should be noted that some online algorithms have to be pre-trained offline and some algorithms referred to as offline can be used sequentially. This depends on the underlying methods used. An offline algorithms' purpose is not to be used online. Nevertheless, online algorithms can be used offline.

TABLE 1. Algorithms used for time-series clustering comparison.

algorithm	type	library	publication
BIRCH[1]	hierarchical	sklearn [15]	[16]
BOCPD[2]	distribution-based	kats	[17]
CluStream[3]	density-based	river [18]	[19]
DBSTREAM[3]	density-based	river [18]	[20]
DenStream[3]	density-based	river [18]	[21]
MiniBatchKMeans[1]	distance-based	sklearn [15]	[22]
STREAMKMeans[3]	distance-based	river [18]	[18]

3) DEPENDENT – INDEPENDENT TSD (TIME VARIANT – TIME INVARIANT)

Depending on the field of study the specific terminology of time dependency can differ. We want to emphasize on the common accepted assumption "An intrinsic feature of a time-series is that, typically, adjacent observations are dependent" ([3, S. 1]). Time dependency characterizes TSD, where a consecutive observation has some connection with its predecessor. In some fields a connection is not necessarily given for two data points in a database that are the closest to each other regarding their timestamp. The dependency of adjacent observations is self-evident, when collecting sensor values of a mechatronic system from an experimental rig, for example. We therefore use the term "time dependency" in the context of a dynamical system and extend it to a time-variant system in the terminology of control systems engineering. Independent TSD would be where the variance and the average are invariant along the time (stationary) and ergodic.

We position our work in the subcategory of **online clustering of dependent multivariate time-series data**. As of now we refer to a time-series as a sequence of dependent observations with a constant sample-rate. A "discrete series of consecutive data points" as a subset of this time-series is synonymously referred to as *pattern, motif, sequence, operating state, state change, between change points, subsequence, episode* or *segment*, among others. In this paper we will use the term ***subsequence*** (using terminology of [10]).

B. ALGORITHMS AND DATASETS

In the relevant literature a diverse number of clustering algorithms can be found [11]. Due to this fact, most of the existing literature for reviewing or surveying existing approaches, attend to a higher-level scope [12], [13], [14]. Fewer are concentrating on time-series clustering [23], [24], [25], online [26], temporal knowledge discovery [5], sequential pattern recognition [10], high dimensional data [27] or change point detection [14], [28]. Current approaches use deep learning architectures like multilayer

[1] https://scikit-learn.org/stable/modules/classes.html
[2] https://facebookresearch.github.io/Kats/
[3] https://riverml.xyz

perceptron (MLP), convolutional neural network (CNN), deep belief network (DBN), generative adversarial network (GAN) and variational autoencoder (VAE) among others [29].

The algorithms we use for comparison are listed in Table 1. All of these are online clustering algorithms that can be used for time-series clustering. Implementations are publicly available in the Python programming language (see *library* column in Table 1) and are well established and tested.

The Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [16] algorithm is based on a clustering features (CF) tree with the CF as a triple of the number of data points, linear sum and the squared sum. This CF tree is built dynamically. It was also one of the earliest algorithms capable of online clustering. The Bayesian Online Change-point Detection (BOCPD) [17] algorithm, as the name suggests, uses Bayesian methods to detect change-points (CPs) online. Since this algorithm only detects CPs, we manipulated the result to be able to interpret every CP as the beginning of a new cluster. This algorithm starts in our comparison with the limitation of not being able to recognize a previously seen cluster. The Stream Clustering Framework (CluStream) [19] algorithm is based on extended CF from BIRCH, following a *k*-means algorithm. The Density-based Stream Clustering (DBSTREAM) [20] algorithm is based on the Self Organizing density-based clustering over data Stream (SOStream) [30] and uses a shared density graph to capture the density between micro-clusters. The Density-Based Clustering over an Evolving Data Stream with Noise (DenStream) [21] algorithm is an extension of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [31] which uses a damped window model of CF to create core-micro-clusters and outlier-micro-clusters. The Mini-Batch K-Means (MiniBatchKMeans) [22] algorithm proposes "the use of mini-batch optimization for k-means clustering" ([22]) to improve the *k*-means optimization problem. The STREAMKMeans [18] algorithm uses an adaption of the original STREAM algorithm from [32]. Replacing the *k*-median subroutine LSEARCH by an incremental *k*-means algorithm. More information and comparison of most of the used algorithms can be found in [26] and [33]

As described in section I, the focus of this publication is on the online multivariate time dependent subsequence clustering using RNN based AE. The number of algorithms within this scope is limited compared to the number of clustering algorithms in general. The following approaches use at least some of those prerequisites. Reference [34] emphasizes the term *segmentation* for an offline sliding window and bottom-up algorithm. Others are converting the time-series into a Markov chain (MC) and then using a Bayesian method to cluster the MCs [35], referring to them as *episodes*. Here the data needs to be discretized into bins of equal length. Reference [36] uses manually selected characteristics (e.g., kurtosis, skewness and frequency) for clustering univariate TSD. Others are using the Augmented Dickey-Fuller test to evaluate time-series stationarity and perform a segmentation based on this [37]. In [38] dynamic latent variables from a vector autoregression (VAR) model in combination with a principal component analysis (PCA) is used for segmenting industrial TSD. Reference [39] shows the advantage of an embedding approach as well, by introducing a PCA and a Vanilla-AE CP detection method with the restriction of focusing on multivariate power grid data.

Focused on transfer learning, [40] introduces an adversarial approach for domain adaption using a stacked AE. Offline convolutional sparse AE used for supervised sequence classification was done by [41] and adapted by [42] for unsupervised motif mining. Other AE based papers are, for example, by [43] who use a mixture of AEs for image and text clustering. Stacked AE and *k*-means for offline clustering is done by [44] without considering time dependency. Showing the combination of GRU-based AE and MTS for anomaly detection is done by [45]. Reference [46] applies the sliding windows approach on CNN-based AE for Anomaly Detection of Industrial Robots. A similar approach using AE for MTS segmentation is published in [47]. The focus there is on change point detection using latent space variables only and no clustering or identification of the subsequences is done.

Clustering is strongly depending on the data and task provided: "…; each new clustering algorithm performs slightly better than the existing ones on a specific distribution of patterns." ([8, S. 268]). Therefore, we try to apply the algorithms on multiple different MTS datasets and compute different metrics for comparison. Large efforts are made for making datasets available to the scientific community and the public to improve comparability and reproducibility by universities or governmental institutions [48], [49]. For this paper we focus on data with multivariate quantitative features with continuous values. For a list of the datasets see Table 3 and a brief description is given in section IV. Evaluating the performance of a clustering algorithm can be done with two different approaches. If external knowledge about the ground truth of each data point and its cluster is known, then so-called external measures can be applied. If no ground truth is available, internal measures need to suffice. Many external measures exist, like the well-known F1-score (based on the effectiveness measure by [50]). With the large number of data available and working in the context of transient machine behavior with the focus on finding internal states of the system, acquiring or providing the ground truth is time-consuming, error-prone and cumbersome (as described in section I). "The definition of clusters depends on the user, the domain, and it is subjective." ([25, S. 30]). We therefore use internal measures for comparing our approach. Those internal measures commonly rely on a similarity measure of the actual data which is being clustered. Thorough work on metric comparison and similarity measures has been done [25], [51], [52]. Most of those measures are based on simple distances and densities computed for each data point but do not take time dependency into consideration. Because of

this, we found that for the use case described in this paper, the commonly used clustering evaluation measures are not well suited for "time-series clustering evaluation measures". In section V we introduce an approach for similarity measures which considers time dependency in combination with well-established clustering metrics (see Table 2).

TABLE 2. Metrics used for time-series clustering comparison. Implementations used from [15].

metric	valuation	value
silhouette [53]	Ratio of distance to its own cluster and distance to the nearest cluster center	$\{s \in \mathbb{R} : -1 \leq s \leq 1\}$ the higher, the better
calinski-harabasz [54]	Ratio of between-cluster variance and the within-cluster variance	$\{s \in \mathbb{R} : 0 \leq s \leq \infty\}$ the higher, the better
davies-bouldin [55]	Ratio of cluster size and between-cluster distance	$\{s \in \mathbb{R} : 0 \leq s \leq \infty\}$ the lower, the better

TABLE 3. Datasets used for time-series clustering comparison.

dataset	type	features
bee-waggle[4] [56]	feature extraction from video	4
cmapss[5] [57]	simulation	18
eigen-worms[6] [58]	feature extraction from video	6
hydraulic[7] [59]	test rig sensors	17
lorenz-attractor [60]	computation	3
mocap[8] [61]	motion capturing sensors	93
occupancy[9] [62]	measurement	5
thomas-attractor [63]	computation	3

III. DEFINITIONS AND RESTRICTIONS

In the following section we define in more detail our data, together with the restrictions of our environment. Considering online clustering, we can refer to our TSD as continuously incoming data or streaming data. This data is considered multivariate when the dimension (number of sensors or features) of the data stream $d > 1$. When we denote one value of one feature as x, we have at time step t the following feature vector:

$$\boldsymbol{x}_t = (x_0, x_1, \ldots, x_d)^T \quad \text{with } x \in \mathbb{R} \text{ and } d \in \mathbb{N} \tag{1}$$

whereas the natural numbers include zero $\{0, 1, 2, \ldots\} = \mathbb{N}$. A complete measurement sequence with n number of time steps using $\boldsymbol{x}$ from Equation (1) as

$$X = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \quad \text{with } n \in \mathbb{N} \tag{2}$$

so, $X \in \mathbb{R}^{d \times n}$. With time dependency consideration, it is reasonable to denote a sliding window of the streaming data, considering Equation (1) and n the number of samples already collected as:

$$W_t = \left(\boldsymbol{x}_{t+0}, \boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t+\zeta}\right) \Rightarrow (\zeta \in \mathbb{N}) \wedge (\zeta < n) \tag{3}$$

Let's also assume, that within the measurement data X there exist subsequences S_j which satisfy our requirements of non-overlapping and variable length. For the indexes of our subsequences, we denote

$$\mathcal{J} = \{j \in \mathbb{N} : j \leq u\} \tag{4}$$

where u is the number of identified subsequences in X. The subsequence then is a continuous sampling from X for a small period of time steps with consecutive data points and the length of m. The subsequence length is usually much smaller than the length of the full measurement data $m \ll n$.

$$S_j = \left(\boldsymbol{x}_{q_j}, \ldots, \boldsymbol{x}_{q_j+m_j}\right) \quad (0 \leq q \leq n - m) \wedge j \in \mathcal{J} \tag{5}$$

For each subsequence with the index j we have a first time step index $q_j = q_{j,start}$ and a last $q_j + m = q_{j,end}$ for which we do not allow overlapping

$$\forall j \in \mathcal{J} \quad \exists (q_{j,start}, q_{j,end}) \\ \Rightarrow (q_{j,start} < q_{j,end}) \wedge ((q_{j-1,end} < q_{j,start}) \wedge j > 0) \tag{6}$$

This results in our uniquely identified non-overlapping set of subsequences

$$\mathcal{S} = \{S_0, \ldots, S_j\} \quad j \in \mathcal{J} \tag{7}$$

Clustering these uniquely identified subsequences results in recognizing reoccurring subsequences and combining them into a subset of all subsequences

$$C_i \subseteq \mathcal{S} \tag{8}$$

which results in the following cluster set $\mathcal{C}$

$$\mathcal{C} = \{C_0, \ldots, C_i\} \quad i \in \mathcal{I} \tag{9}$$

with the cluster index or unique cluster label

$$\mathcal{I} = \{i \in \mathbb{N} : i < n\} \tag{10}$$

For the output of a clustering algorithm at time t we denote the scalar value y_t as our label or designated subsequence identification. For evaluation purposes a clustering for a time-series produces a label array $\boldsymbol{y}$ for all time steps:

$$\boldsymbol{y} = (y_0, y_1, \ldots, y_n) \quad \text{with } y \in \mathcal{J} \text{ and } n \in \mathbb{N} \tag{11}$$

Furthermore, it is a requirement, that the streaming data provided can be applied to a numerical differentiation algorithm. Therefore, a constant sample rate is necessary and in case of strong noise, filtering or smoothing of the data should be applied by a preprocessing step. Also, there mustn't be missing values and extreme outliers need to be removed.

[4] https://sites.cc.gatech.edu/~borg/ijcv_psslds/
[5] https://data.nasa.gov/dataset/C-MAPSS-Aircraft-Engine-Simulator-Data/xaut-bemq
[6] http://www.timeseriesclassification.com/description.php?Dataset=EigenWorms
[7] https://archive.ics.uci.edu/ml/datasets/Condition%20monitoring%20of%20hydraulic%20systems
[8] http://mocap.cs.cmu.edu/
[9] https://github.com/LuisM78/Occupancy-detection-data

In our use case we assume that some knowledge about the incoming data exists, so that an estimate of the variance and the mean of the variable can be performed for standardization.

IV. DATASETS

All datasets used for comparison in this work are described briefly in this section and listed in Table 3. They all contain quantitative features with continuous values. For further use of the datasets, no missing values exist, the data is continuous and was standardized for the algorithms but not for the metric computations. No other preprocessing like smoothening or filtering was performed.

The **bee-waggle** dataset [56] contains movement of bees in a hive captured with a vision-based tracker. The first two features are the x and y coordinates of the bee added with the sine and the cosine function applied to the heading angle.

The **cmapss** dataset is a ''dataset of run-to-failure trajectories for a small fleet of aircraft engines under realistic flight conditions'' [57] with 18 features.

The **eigen-worms** dataset [58] contains measurements of worm motion. Preprocessing extracted six features, which represent the amplitudes along six previously identified base shapes of the worms

The **hydraulic** dataset [59] is obtained from a hydraulic test rig with measuring 17 process values such as pressures, volume flows and temperatures.

Lorenz-attractor refers to a synthetic dataset which is calculated using a system of the three coupled ordinary differential equations which represent a hydrodynamic system: $\dot{X} = s(Y - X); \dot{Y} = rX - Y - XZ; \dot{Z} = XY - bZ$ with parameters used $s = 10, r = 28$ and $b = 2.667$ (see Figure 2). ''In these equations X is proportional to the intensity of the convective motion, while Y is proportional to the temperature difference between the ascending and descending currents, similar signs of X and Y denoting that warm fluid is rising, and cold fluid is descending.'' [60]

The **mocap** or The Motion Capture Database (MOCAP) dataset [61] contains 93 features from human motion captured with markers.

The **occupancy** dataset [62] is a measurement of sensory data in an office with the following sensors: temperature, humidity, the derived humidity ratio, light and CO2.

The **thomas-attractor** dataset is as the lorenz-attractor dataset, a synthetic dataset, computed with the three coupled differential equations: $\dot{X} = \sin(Y) - bX; \dot{Y} = \sin(Z) - bY; \dot{Z} = \sin(X) - bZ$ originally proposed by [63] with the parameter used $b = 0.1615$.

V. MULTIVARIATE TIME-SERIES SUB-SEQUENCE CLUSTERING METRIC (MT3SCM)

As emphasized in section I and section II, to our knowledge, none of the existing clustering metrics take into consideration the time space variations like curvature, acceleration or torsion in a multidimensional space. We believe using these curve parameters, is an intuitive method to measure similarities between mechatronic system state changes or

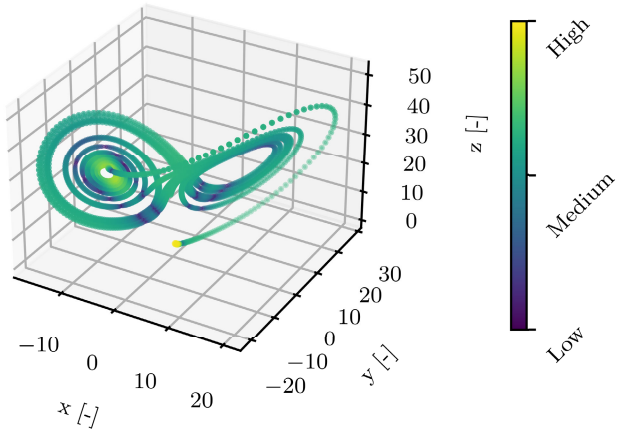

FIGURE 2. Lorenz-attractor dataset. Computed with $\dot{X} = s(Y - X)$; $\dot{Y} = rX - Y - XZ$; $\dot{Z} = XY - bZ$ and parameters used $s = 10$, $r = 28$ and $b = 2.667$. Color and marker size indicate amount of curvature on a logarithmic scale for better visibility.

subsequences in MTSD in general (in regard to the restrictions in section III).

Our *MT3SCM* score consists of three main components.

$$mt3scm = (cc_w + s_L + s_P)/3 \tag{12}$$

The weighted curvature consistency (cc_w), the silhouette location based (s_L) and the silhouette curve-parameter based (s_P). When making the attempt of clustering TSD, it is subjective and domain specific. Nevertheless, we try to take the intuitive approach of treating MTSD as space curves and use the parameterization as a similarity measure. This is done in two different ways. First, we create new features by computing the curve parameters sample by sample (e.g., curvature, torsion, acceleration) and determine their standard deviation for each cluster. Our hypothesis is, that with a low standard deviation of the curve parameters inside a cluster, the actions of a mechatronic system in this cluster are similar. We call this the curvature consistency (cc) (see Equation (24) used in 14 in algorithm 1). The second procedure is to apply these newly computed features, which are computed to scalar values per subsequence, onto a well-established internal clustering metric, the silhouette score [53] (see Table 2).

The computation of the cc comprises the calculation of the curvature κ and the torsion τ at every time step t with $\boldsymbol{x}_t$.

$$\kappa(t) = \frac{\langle \dot{\boldsymbol{e}}_1(t), \boldsymbol{e}_2(t) \rangle}{\|\dot{\boldsymbol{x}}_t\|} \tag{13}$$

$$\tau(t) = \frac{\langle \dot{\boldsymbol{e}}_2(t), \boldsymbol{e}_3(t) \rangle}{\|\dot{\boldsymbol{x}}_t\|} \tag{14}$$

whereas $\boldsymbol{e}_1$ is the unit tangent vector (or first Frenet vector), $\boldsymbol{e}_2$ is the unit normal vector (or second Frenet vector) and $\boldsymbol{e}_3$ is the unit binormal vector (or third Frenet vector) which are

defined as:

$$\boldsymbol{e}_1(t) = \frac{\dot{\boldsymbol{x}}_t}{\|\dot{\boldsymbol{x}}_t\|} \tag{15}$$

$$\bar{\boldsymbol{e}}_2(t) = \ddot{\boldsymbol{x}}_t - \langle \ddot{\boldsymbol{x}}_t, \boldsymbol{e}_1(t) \rangle \times \boldsymbol{e}_1(t) \tag{16}$$

$$\boldsymbol{e}_2(t) = \frac{\bar{\boldsymbol{e}}_2(t)}{\|\bar{\boldsymbol{e}}_2(t)\|} \tag{17}$$

$$\bar{\boldsymbol{e}}_3(t) = \dddot{\boldsymbol{x}}_t - \langle \dddot{\boldsymbol{x}}_t, \boldsymbol{e}_1(t) \rangle \times \boldsymbol{e}_1(t) - \langle \dddot{\boldsymbol{x}}_t, \boldsymbol{e}_2(t) \rangle \times \boldsymbol{e}_2(t) \tag{18}$$

$$\boldsymbol{e}_3(t) = \frac{\bar{\boldsymbol{e}}_3(t)}{\|\bar{\boldsymbol{e}}_3(t)\|} \tag{19}$$

From which we can also derive the speed $v = \|\dot{\boldsymbol{x}}_t\|$ and the acceleration $a = \|\ddot{\boldsymbol{x}}_t\|$. Figure 3 shows exemplarily the curvature κ, torsion τ, speed v and acceleration a for the first part of the thomas-attractor dataset.

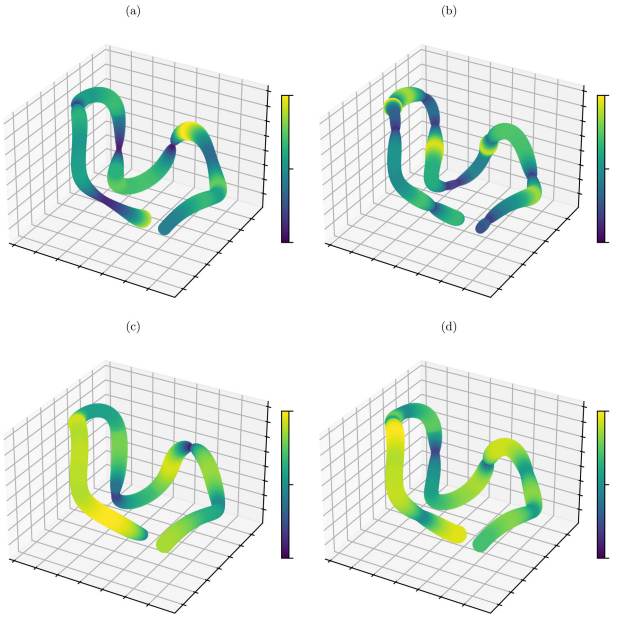

FIGURE 3. Qualitative visualization of the (a) curvature κ, (b) torsion τ, (c) speed v and (d) acceleration a computed on part of the thomas-attractor dataset. Color and marker size indicate amount of curve parameter on a logarithmic scale for better visibility (dark and thin means low value, bright and thick means high value). Axis labels and colorbar labels are along the lines of Figure 2.

Afterwards the cc is calculated per cluster $i \in \mathcal{I}$, by taking the empirical standard deviation for each curve parameter (exemplarily for κ in Equation (20) with the set of subsequence indexes $\mathcal{J}_i$ within our cluster i). The arithmetic mean (Equation (21)) of the standard deviations for the curvature κ, torsion τ and the acceleration a results in the final cc per cluster (see Equation (22)).

$$\sigma_{\kappa_i} = \sqrt{\frac{1}{N_i - 1} \sum_{j \in \mathcal{J}_i} \sum_{n=q_j}^{q_j+m_j} (\kappa_n - \bar{\kappa}_i)^2} \tag{20}$$

$$\sigma_i = \frac{\sigma_{\kappa_i} + \sigma_{\tau_i} + \sigma_{a_i}}{3} \tag{21}$$

$$cc_i = 1 - \sigma_i \quad \text{with } cc_i \in \mathbb{R} : cc_i \leq 1 \tag{22}$$

$$cc_i = \begin{cases} cc_i, & \text{if } cc_i > -1 \\ -1, & \text{if } cc_i \leq -1 \end{cases} \tag{23}$$

The cc_w, is directly derived from the cc per cluster, by weighting it with the number of data points per cluster $i \in \mathcal{I}$ Equation (24).

$$cc_w = \frac{\sum_{i=1}^{n} cc_i \times N_i}{\sum_{i=1}^{n} N_i} \tag{24}$$

The calculation of the scores s_P and s_L is different to the standard estimation of the silhouette score, which is shown in Equation (25) and originally based on every data point of the time-series X and the assigned cluster label array y:

$$s = f(X, \boldsymbol{y}) \tag{25}$$

Our s_P is the silhouette score derived from our previously computed curve parameters per subsequence per cluster as well as the standard deviation of those and the number of data points per subsequence.

$$s_P = f(\widetilde{X}_{s_P}, \boldsymbol{y}_j) \tag{26}$$

with

$$\widetilde{X}_{s_P} = \begin{pmatrix} \bar{\kappa}_{11} & \bar{\tau}_{11} & \bar{a}_{11} & \sigma_{11} & N_{11} \\ \bar{\kappa}_{12} & \bar{\tau}_{12} & \bar{a}_{12} & \sigma_{12} & N_{12} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{\kappa}_{21} & \bar{\tau}_{21} & \bar{a}_{21} & \sigma_{21} & N_{21} \\ \bar{\kappa}_{22} & \bar{\tau}_{22} & \bar{a}_{22} & \sigma_{22} & N_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{\kappa}_{ij} & \bar{\tau}_{ij} & \bar{a}_{ij} & \sigma_{ij} & N_{ij} \end{pmatrix}, \boldsymbol{y}_j = \begin{pmatrix} y_{11} \\ y_{12} \\ \vdots \\ y_{21} \\ y_{22} \\ \vdots \\ y_{ij} \end{pmatrix} \tag{27}$$

The s_L uses the silhouette score based on the median value $\hat{x}_{d_{ij}}$ of a subsequences original feature space per feature d.

$$s_L = f(\widetilde{X}_{s_L}, \boldsymbol{y}_j) \tag{28}$$

with

$$\widetilde{X}_{s_L} = \begin{pmatrix} \hat{x}_{1_{11}} & \hat{x}_{2_{11}} & \dots & \hat{x}_{d_{11}} & \sigma_{11} & N_{11} \\ \hat{x}_{1_{12}} & \hat{x}_{2_{12}} & \dots & \hat{x}_{d_{12}} & \sigma_{12} & N_{12} \\ \vdots & & & & \vdots & \vdots \\ \hat{x}_{1_{21}} & \hat{x}_{2_{21}} & \dots & \hat{x}_{d_{21}} & \sigma_{21} & N_{21} \\ \hat{x}_{1_{22}} & \hat{x}_{2_{22}} & \dots & \hat{x}_{d_{22}} & \sigma_{22} & N_{22} \\ \vdots & & & & \vdots & \vdots \\ \hat{x}_{1_{ij}} & \hat{x}_{2_{ij}} & \dots & \hat{x}_{d_{ij}} & \sigma_{ij} & N_{ij} \end{pmatrix} \tag{29}$$

The main idea of this approach is to combine three main parts inside one metric. First incentive is to reward a **low standard deviation of the curve parameters** in between a cluster (accomplished by cc). Second, to benchmark the clusters **spatial separation based on the new feature space** (curve parameters, accomplished by s_P). And third, to benchmark the clusters **spatial separation based on the median of the**

subsequence in the original feature space (accomplished by s_L). The proposed algorithm for this new metrics computation is described in algorithm 1.

Algorithm 1 *MT3SCM*

procedure MT3SCM(X, y) ▷ Data $X \in \mathbb{R}^{d \times n}$ and labels $\boldsymbol{y} \in \mathbb{N}^n$
  $L \leftarrow empty()$ ▷ Array initialization for all subsequence median coordinates or **L**ocation
  $P \leftarrow empty()$ ▷ Array initialization for all subsequence curve **P**arameters mean values
  $K \leftarrow$ GetCurveParametersForAllData(X)
  $\boldsymbol{y}_{unique} \leftarrow$ FindUniqueClusterIDs($\boldsymbol{y}$)
  for i in $\boldsymbol{y}_{unique}$ **do**
    $X_i \leftarrow$ GetClusterData(X, i)
    $\boldsymbol{s} \leftarrow$ FindSubsequences(y, i)
    for j in $\boldsymbol{s}$ **do**
      $X_{i,j} \leftarrow$ GetSubsequenceData(X_i, j)
      $L[i, j] \leftarrow$ GetMedianLocations($X_{i,j}$)
      $P[i, j] \leftarrow$ GetCurveParameterValues(K, i, j)
    end for
    $cc_i \leftarrow$ ClusterCurvatureConsistency(P) ▷ Compute the cluster curvature consistency (cc_i) with the empirical standard deviation of each curve parameter over time. If the cluster consists only of one time step, set the cc_i to zero.
    $C[i] \leftarrow cc_i$ ▷ Collect cc_i data for all clusters
  end for
  $cc_w \leftarrow$ WeightedAverage(C, n_{pc}) ▷ Compute weighted average curvature consistency (cc_w) from cc_i with number of points per cluster
  $s_L \leftarrow$ SilhouetteComputation(L, $\boldsymbol{y}_{unique}$) ▷ Compute the silhouette coefficient using the center positions of each identified subsequence
  $s_P \leftarrow$ SilhouetteComputation(P, $\boldsymbol{y}_{unique}$) ▷ Compute the silhouette coefficient with the curve parameters
  $score \leftarrow (cc_w + s_L + s_P)/3$
  return *score* ▷ The final score
end procedure

A. EVALUATION

For computational tests, we manually created a "perfect" synthetic dataset with respect to our metric (see Figure 4). Figure 4 (a) shows the original synthetic dataset, where the subsequences in cluster 1 are a helix along the increasing x axis. For cluster 2 the subsequences are a straight movement, with quadratic decreasing distances along the y axis. Cluster 3 is representing a helix along the decreasing x axis but with a different resolution than cluster 1. Cluster 4 is, along with cluster 2, a straight movement with quadratic increasing distances along the y axis. This cycle is repeated six times. Figure 4 (b) shows the new feature space for the s_L component. The feature space for the s_P component is shown in Figure 4 (c). Applying the new features per subsequence on the standard metrics, results in best scores for all metrics. This shows that the new feature space allows a good separation in contrast to the original space, as proven by the metrics scores for silhouette, calinski-harabasz and davies-bouldin on the original and the two new feature spaces. To show the benefit of the new feature space, we applied the agglomerative clustering[10] not on the original lorenz-attractor dataset but on the newly computed feature space based on curvature, torsion and acceleration (see Figure 5) The metric values for Figure 5 (b) show a high cc_w and a decent s_P value for the low number of 10 clusters specified.

To further evaluate our metric, we used the lorenz-attractor and the thomas-attractor dataset (see Table 3) and applied an agglomerative clustering, a time-series k-means clustering as well as a random subsequence clustering. Varying the number of clusters and some algorithm specific parameters. Afterwards the metrics calinski-harabasz, davies-bouldin and silhouette scores were computed and compared to our new metric *MT3SCM*. From these results we derived a correlation matrix (see Figure 6). The cc and the cc_w are clearly related due to their direct combination. The positive correlation between the internal components to the overall *MT3SCM* score is obvious. We see a clear positive correlation to the silhouette score which is evident due to the internal use of this metric. Interestingly, the correlation between the cc_w and the s_P is negative. This is due to the types of datasets and algorithms we used. Because with higher number of clusters we theoretically expect a better cc because of the lower standard deviation by chance. On the other hand, the more clusters exist, the more likely a similar curve parameter between the clusters exists and therefore creates a new feature space with overlapping clusters, which results in a low s_P score. This can be retraced within the subfigures of Figure 7. The low correlation between the calinski-harabasz and the davies-bouldin scores supports our point that the available clustering metrics are not well suited to be used for time-series clustering evaluation measures. Figure 7 shows examples where the agglomerative clustering was applied on the lorenz-attractor dataset (part of the data used for the correlation matrix Figure 6). It can be seen that the agglomerative clustering on the original dataset is not an optimal cluster algorithm, when comparing the metrics to Figure 5 (b). Comparing Figure 5 (b) and Figure 7 (d) we can see a similar *MT3SCM* score but very different standard metrics scores. The similar *MT3SCM* score is based on the much higher number of clusters and equally distributed subsequence length in Figure 7 (d), which results in a high cc_w value as well as a good spatial separation (s_L), which is compensating the low s_P value due to the similar curve parameters of the clusters. Figure 5 (b) however, also has a very high cc_w value with a good s_P value reaching a similar *MT3SCM* score but with a fifth of the number of clusters. How our metric handles random clustering with

[10] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

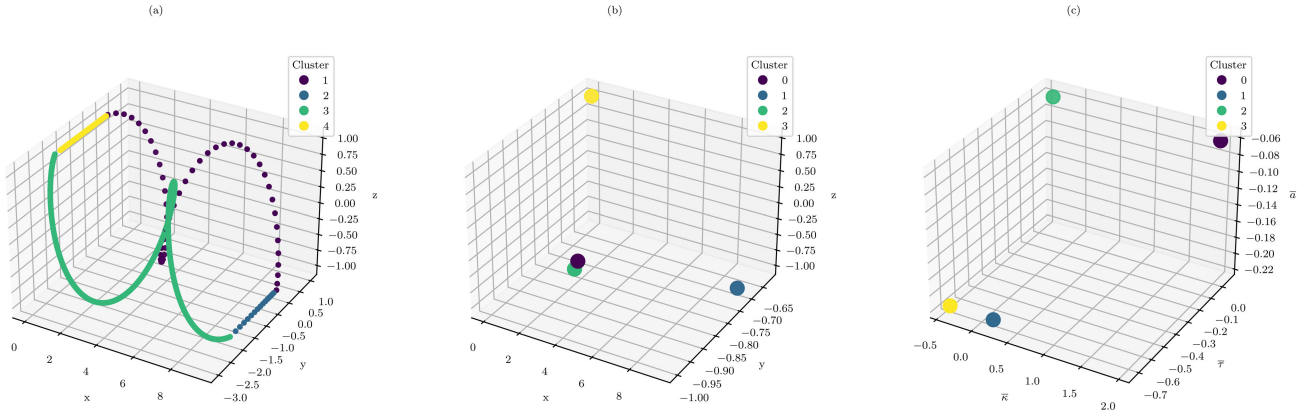

FIGURE 4. Synthetic dataset with four clusters with a perfect own metric score of $mt3scm = 1$ due to each cluster's unique and constant curve parameters. (a) Synthetic dataset with best own result of $mt3scm = 1$. Standard metrics scores computed with original data; davies-bouldin: 1.4, calinski-harabasz: $6.9e+02$, silhouette: 0.087. (b) New feature space from the centers (median value) of each subsequence. Standard metrics scores computed with new feature space; davies-bouldin: $6.3e-07$, calinski-harabasz: $3.8e+14$, silhouette: 1. (c) New feature space from the curve parameters extracted from each subsequence. Standard metrics scores computed with new feature space; davies-bouldin: $6.8e-07$, calinski-harabasz: $1.2e+13$, silhouette: 1.

TABLE 4. Metric values for Figure 7 8 and 5.

Figure	davies-bouldin	calinski-harabasz	silhouette	*MT3SCM*	cc	cc_w	s_L	s_P
Figure 5 (a)	0.64	4.5e+03	0.51	0.1	0.51	0.29	0.045	-0.026
Figure 5 (b)	13	86	-0.32	0.23	0.65	0.77	-0.19	0.11
Figure 5 (c)	0.71	2.3e+02	0.43	0.13	0.4	0.4	0	0
Figure 5 (d)	63	9.1	-0.3	0.18	0.36	0.54	0	0
Figure 7 (a)	0.74	6.6e+02	0.34	0.071	0.24	0.22	0.02	-0.029
Figure 7 (b)	1.1	1.9e+03	0.32	0.027	0.25	0.23	-0.013	-0.13
Figure 7 (c)	0.85	1.1e+03	0.29	0.034	0.54	0.42	-0.042	-0.28
Figure 7 (d)	0.78	2.4e+03	0.33	0.26	0.78	0.73	0.37	-0.33
Figure 8 (a)	70	0.49	3.4E-05	-0.00092	-0.00096	-0.00089	-0.0012	-0.00068
Figure 8 (b)	7	48	0.029	-0.053	0.092	0.078	-0.12	-0.12
Figure 8 (c)	65	1.6	5.2E-05	-0.00047	-0.00084	-0.00083	-0.00037	-0.00022
Figure 8 (d)	3.5	5.4E+02	0.039	-0.0035	0.00041	0.0011	-0.0025	-0.0091

critical scenarios, is shown in Figure 8. The Python code and a more detailed evaluation are publicly available at [64]

B. CONCLUSION

We have described a more suitable similarity measure for dependent TSD. After showing how to compute our metric and evaluated on different datasets its use case and effectiveness. Further we will use this metric in addition to the standard metrics to evaluate our proposed online time-series clustering algorithm which is described in section VI

VI. CLUSTERING ALGORITHM (ABIMCA)

In this section we describe the concept of our time-series clustering approach in detail. Afterwards, we apply our algorithm onto the datasets described in section IV and present the results.

A. METHOD

As described in [23] a key component in a time-series clustering algorithm is the similarity function to quantify the clustering criteria. Common similarity functions used are distance measures like euclidean distance or some kind of correlation coefficients like Pearson's correlation coefficient. Those are also used for static data clustering algorithms. More suitable for time-series clustering are similarity functions like Dynamic Time Warping (DTW) distance, short time-series (STS) distance [65] or considering space curves like we introduced in section V.

In this work we analyzed an approach which is data driven, based on unsupervised machine learning algorithms and has online capabilities (see Figure 11). Our approach uses a RNN based AE to generate scores which are used as similarity measures. Specifically, the experiments in this work were performed using a pytorch [66] implementation of a bidirectional one-layer gated recurrent unit (GRU) RNN with a hidden size of the input dimensions minus one $h = d - 1$. Other prerequisites regarding the dataset and preprocessing are described in section IV and section III.

The main procedure of the approach is as follows: The incoming data is taken as a sliding window W_t at the current

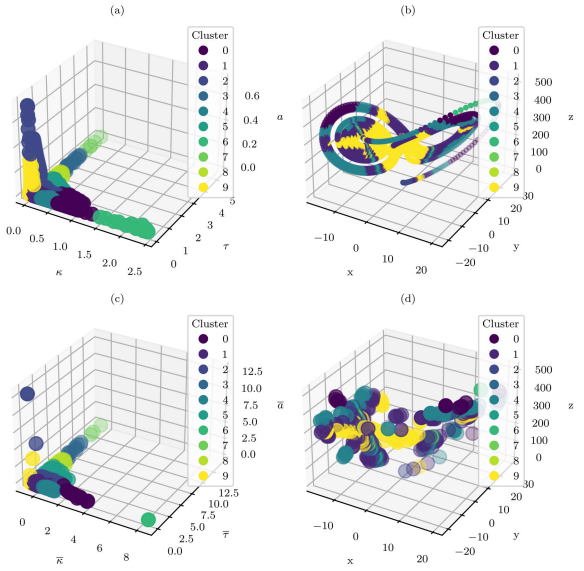

FIGURE 5. Lorenz-attractor dataset with 10 clusters from agglomerative clustering on the new curve parameters feature space. See Table 4 for metric comparison of the following subplots. (a) New curve parameters feature space computed from the Lorenz-attractor dataset with labels from agglomerative clustering (b) Lorenz-attractor dataset with labels from agglomerative clustering on the new curve parameters feature space (c) New feature space from the curve parameters extracted from each subsequence. Standard metrics scores computed with new feature space (d) New feature space from the centers (median value) of each subsequence. Standard metrics scores computed with new feature space.

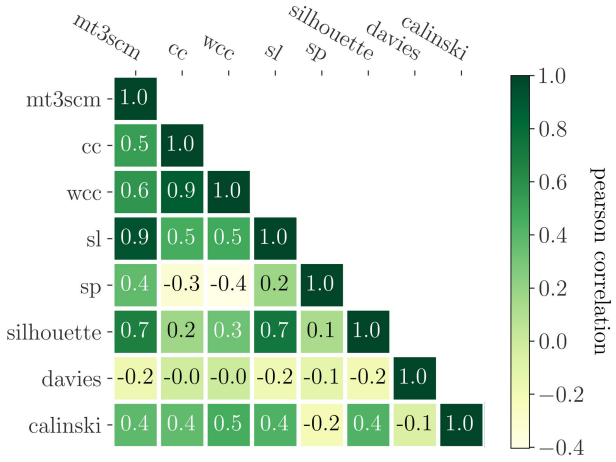

FIGURE 6. Own metric (*MT3SCM*) correlation analysis. Own metric and its four subcomponents (curvature consistency (*cc*), weighted curvature consistency (cc_W), silhouette location based (s_L), silhouette curve-parameter based (s_P)) correlation to calinski-harabasz, davies-bouldin and silhouette score for random, agglomerative and k-means clustering on lorenz and thomas-attractor dataset.

time t with length ζ of past time steps and number of features d. This matrix $W_t \in \mathbb{R}^{d\times\zeta}$ is used for the input of, what we call, the Base Autoencoder (BAE). The key element of our algorithm is, that this BAE's parameters are not constant but being adapted iteratively with a stochastic gradient descent (SGD) optimization method for each new incoming sliding window. For this training of the BAE, we use a slight adaption of the sparse AE loss function $\mathcal{L}$ from [67] with a basic

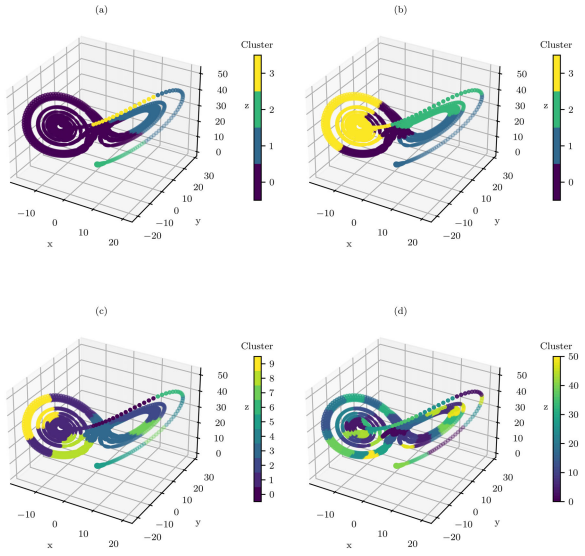

FIGURE 7. Agglomerative clustering from [15] applied on the lorenz-attractor dataset exemplifies the unique components of our metric compared to the silhouette calinski-harabasz and davies-bouldin scores. See Table 4 for metric comparison of the following subplots. Subfigures (a) (b) and (c) all have a similarly low *MT3SCM* score compared to Figure 5 (b) but considerably good standard metric scores. Subfigure (d) can achieve a relatively high *MT3SCM* score due to the high number of clusters and the resulting good cc_W and s_L value which compensates the low s_P value.

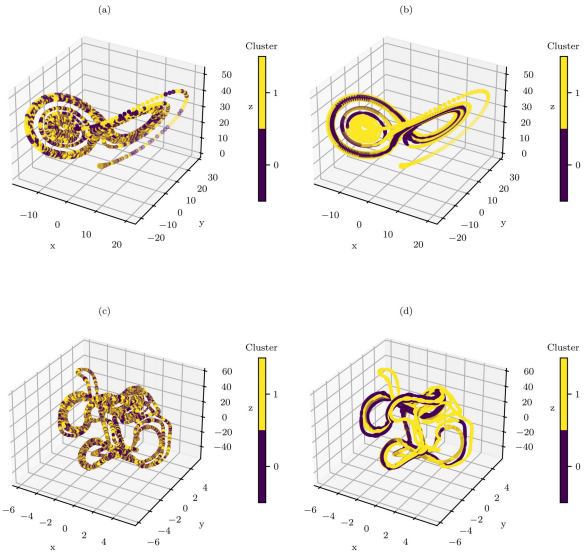

FIGURE 8. Own metric evaluation using random clusterer on thomas-attractor dataset and lorenz-attractor dataset. See Table 4 for metric comparison of the following subplots (a) Own metric and all of its subcomponents are around zero, as desired. Calinski-harabasz value is low and davies-bouldin is high, which also indicate a "bad" clustering (b) Longer random subsequences also generate a *MT3SCM* result around zero. Calinski-harabasz and davies-bouldin scores are stronger influenced by the subsequence length (c) Own metric and all of its subcomponents are around zero, as desired. Calinski-harabasz value is low and davies-bouldin is high, which also indicate a "bad" clustering (d) As seen for the lorenz-attractor data in (b), longer subsequences have a high impact on calinski-harabasz and davies-bouldin scores.

regularization term or sparsity penalty Ω

$$loss = l = \mathcal{L}(W_t, \widetilde{W}_t, \mathbf{h}) = MSE(W_t, \widetilde{W}_t) + \Omega(\mathbf{h}) \quad (30)$$

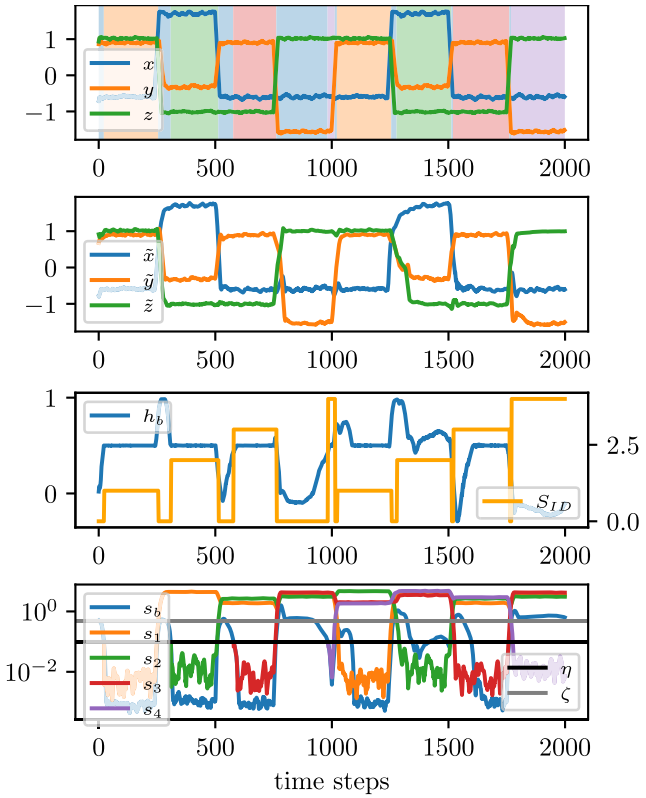

FIGURE 9. Example of online clustering with a simple three-dimensional synthetic dataset. First row shows the original input data X (or the last values of each sliding window W_t) with the online cluster IDs as the background color (blue is unknown or $S_{ID} = 0$). Second row shows the output of the AE or the reconstruction $\widetilde{W}$. In the third row the blue line represents the value of the latent space h_b (left axis) and the identified subsequence ID S_{ID} (right axis). The last row indicates the Base Autoencoder's (BAE) (s_b) as well as the SAEs' (s_1 - s_4) score values. The black horizontal line is the subsequence detection score threshold (η) and the gray line is the subsequence recognition score threshold (ρ).

TABLE 5. Summation of the number of outperformances of each algorithm for all datasets and all metrics compared to the mini-batch-kmeans with parameters from hyperparameter search.

Outperforms		mini-batch-kmeans			Total
Metric	silhouette	calinski-harabasz	davies-bouldin	mt3scm	
Algorithm					
CluStream	0	0	0	3	3
BOCPDetector	0	4	2	1	7
BIRCH	2	3	1	2	8
STREAMKMeans	1	2	3	5	11
DBSTREAM	2	**6**	6	3	17
DenStream	**5**	4	**8**	5	22
ABIMCA	**5**	4	7	**7**	**23**

where $\mathbf{h} = f(\mathbf{x})$ is the encoders output or latent space. The sparsity penalty we denote as:

$$\Omega(\mathbf{h}) = \lambda \cdot \sum_{i=1}^{d-1} |h_i - c_{lc}| \tag{31}$$

with the penalty factor $\lambda = 1e{-}10$ and the latent center constant $c_{lc} = 0.5$. The mean squared error (MSE) is

$$MSE(W_t, \widetilde{W}_t) = \frac{1}{d \cdot \zeta} \sum_{i=1}^{d} \sum_{j=1}^{\zeta} (w_{ij} - \widetilde{w}_{ij})^2 \tag{32}$$

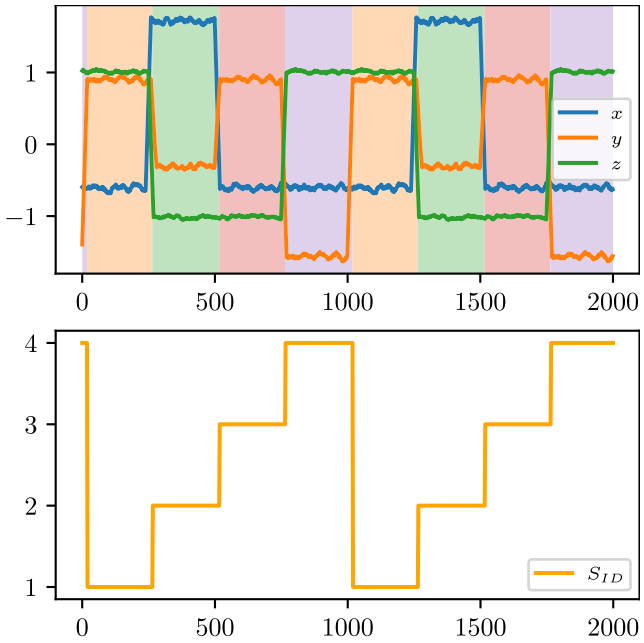

FIGURE 10. Example of batch-wise offline clustering with a simple three-dimensional synthetic dataset.

TABLE 6. Summation of the number of outperformances of each algorithm for all datasets and all metrics compared to the mini-batch-kmeans with default parameters.

Outperforms		mini-batch-kmeans			Total
Metric	silhouette	calinski-harabasz	davies-bouldin	mt3scm	
Algorithm					
CluStream	1	0	0	5	6
BIRCH	3	0	0	5	8
DenStream	1	1	3	**6**	11
ABIMCA	3	1	4	4	12
DBSTREAM	2	4	3	3	12
STREAMKMeans	**5**	1	3	5	14
BOCPDetector	1	**6**	**6**	5	**18**

which results in the final loss computation

$$loss = l = \frac{1}{d \cdot \zeta} \sum_{i=1}^{d} \sum_{j=1}^{\zeta} (w_{ij} - \widetilde{w}_{ij})^2 + \lambda \cdot \sum_{i}^{d-1} |h_i - c_{lc}| \tag{33}$$

where the first part is the MSE between the input matrix W_t and the reconstruction $\widetilde{W}_t$ and the second part is the penalty of the latent space deviation.

To determine if a subsequence is recognized at the current time step, we denote the scoring function SF as follows

$$s = score = SF(l, \mathbf{h}) = c_{fw} \cdot (|c_{lc} - \mathbf{h}|) + \frac{l}{c_{fw}} \tag{34}$$

whereas the weighting factor in current implementation is $c_{fw} = 1$ and latent center constant is $c_{lc} = 0.5$. It utilizes the reconstruction error as well as the deviation of the latent space with a doubled emphasis on the latent space deviation due to its dependence in the loss function as well as the scoring function which includes the loss again (see Equation (34). In combination with a threshold, the score is used to determine when a recognizable subsequence is present.

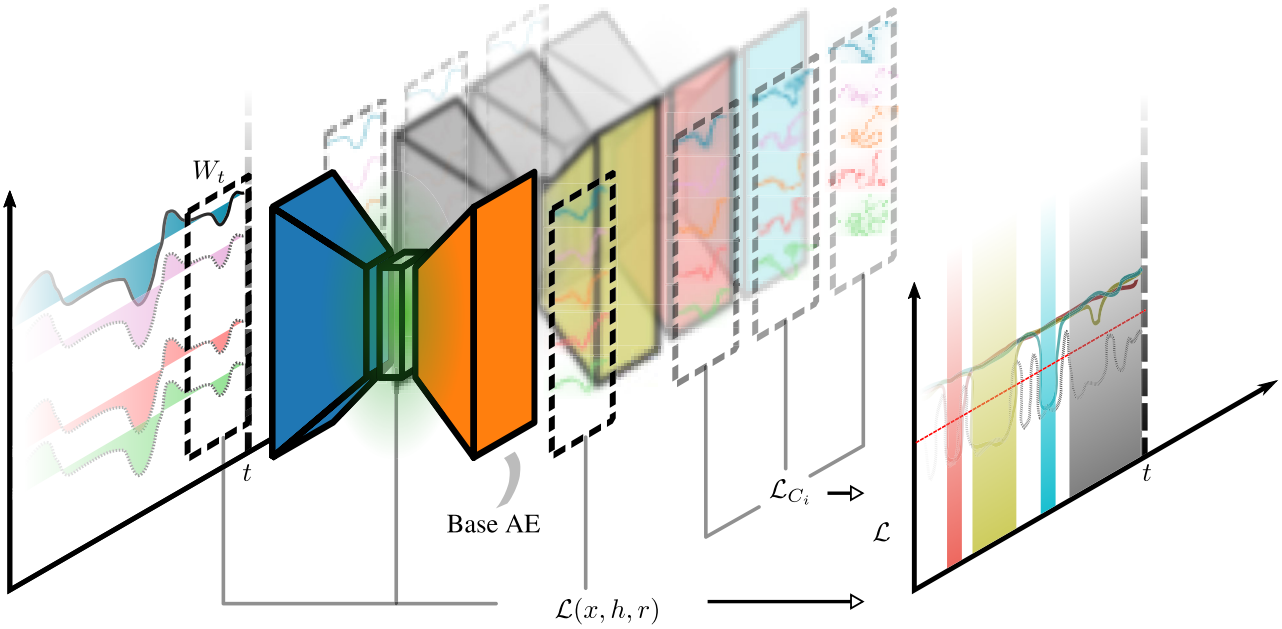

FIGURE 11. Concept of the ABIMCA approach. Sliding window of the MTS W_t is iteratively trained in the base AE. If score of base AE (gray dotted line) is below threshold (dashed red line), a new subsequence AE is created from the base AE. Incoming data is also compared to existing subsequence AEs if subsequence can be recognized.

TABLE 7. Best metric 'metrics.mt3scm' value for each dataset and algorithm from hyperparameter search results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	0.139	0.143	0.125	0.051	0.183	0.075	0.099	**0.370**
cmapss	0.129	0.120	0.213	0.108	0.226	0.141	**0.378**	0.179
eigen-worms	0.018	0.059	0.099	-0.026	**0.389**	0.064	0.099	0.150
hydraulic	0.364	0.104	0.055	0.006	0.590	0.655	**0.667**	0.598
lorenz-attractor	0.242	0.110	0.019	0.272	0.274	0.279	0.272	**0.368**
mocap	0.102	0.047	0.256	**0.277**	0.273	0.257	0.067	0.258
occupancy	0.430	0.214	0.084	**0.697**	0.450	0.267	0.458	0.235
own-synth	0.355	0.159	0.096	0.366	0.279	0.366	0.297	**0.622**
thomas-attractor	0.115	nan	0.013	nan	0.151	0.153	0.079	**0.474**

TABLE 8. Best metric 'metrics.calinski-harabasz' value for each dataset and algorithm from hyperparameter search results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	225.040	169.623	6.085	**1193.353**	357.833	210.572	297.385	324.477
cmapss	2.56e+04	**3.84e+04**	4182.524	2731.331	9154.261	4930.092	1244.679	7847.877
eigen-worms	859.800	2498.356	94.549	**2945.532**	1880.220	1746.451	1145.837	1803.977
hydraulic	1469.136	4612.781	34.909	**1.08e+04**	923.561	3801.743	3350.071	3656.773
lorenz-attractor	992.643	17.658	143.971	580.847	1650.040	**2191.928**	2050.461	1825.448
mocap	445.091	2179.353	148.563	**1.13e+05**	1823.373	2525.665	157.968	1.80e+04
occupancy	5154.791	9190.251	282.272	1.32e+04	**1.63e+04**	7020.677	9255.468	5114.837
own-synth	2255.508	74.232	6.455	**2.29e+04**	1165.953	2257.976	1702.116	1026.830
thomas-attractor	**2012.217**	nan	8.167	nan	1764.227	1859.162	1352.554	1691.685

The bottom row of Figure 9 shows, that a subsequence is present, when the BAE's score (blue line) is below the horizontal black line ($s_b <= \eta$). If a subsequence is present, a copy of the BAE is made and its parameters are frozen and associated with this specific pattern of a subsequence. These copies of the BAE, which we call Subsequence Autoencoder (SAE), are used to recognize previously seen subsequences using the same scoring function. A concept drawing of the approach is shown in Figure 11. The algorithm is described in pseudocode in algorithm 2.

The functionality can be retraced considering Figure 9 and 10. This example shows the algorithm applied

TABLE 9. Best metric 'metrics.davies-bouldin' value for each dataset and algorithm from hyperparameter search results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	71.078	**1.424**	84.739	5.765	14.012	19.554	24.979	12.578
cmapss	40.052	**0.469**	92.083	3.074	46.712	14.087	31.323	16.061
eigen-worms	17.463	**0.866**	21.523	4.476	3.846	2.835	10.923	3.828
hydraulic	68.132	**2.947**	84.968	36.736	152.522	138.098	45.246	31.007
lorenz-attractor	38.516	**0.981**	54.983	14.980	28.773	4.328	20.754	22.631
mocap	35.612	**0.173**	8.030	1.210	3.390	4.812	2.005	17.499
occupancy	65.075	**5.129**	168.704	6.199	20.439	15.437	40.378	23.013
own-synth	27.162	8.29e+04	137.923	3.596	20.822	**2.518**	1895.422	18.762
thomas-attractor	189.853	nan	82.745	nan	5.738	**1.940**	14.247	5.355

TABLE 10. Best metric 'metrics.silhouette' value for each dataset and algorithm from hyperparameter search results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	0.181	-0.104	-0.067	0.092	**0.365**	0.192	0.231	0.318
cmapss	**0.667**	0.007	0.488	0.495	0.636	0.511	0.320	0.570
eigen-worms	0.171	0.025	-0.025	0.108	**0.291**	0.247	0.172	0.272
hydraulic	0.684	-0.272	-0.087	-0.012	0.636	**0.775**	0.769	0.769
lorenz-attractor	0.380	-0.117	0.031	0.270	0.349	0.399	0.387	**0.432**
mocap	0.233	0.020	0.054	0.386	**0.482**	0.429	0.125	0.436
occupancy	0.497	-0.424	-0.184	**0.774**	0.766	0.647	0.598	0.484
own-synth	0.306	-0.233	-0.052	**0.439**	0.346	0.396	0.380	0.383
thomas-attractor	**0.299**	nan	-0.076	nan	0.269	0.282	0.203	0.274

TABLE 11. Metric 'metrics.mt3scm' value for each dataset and algorithm from default calibration results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	-0.009	0.143	-0.021	-0.100	nan	-0.097	0.026	**0.282**
cmapss	-0.104	**0.120**	-0.065	0.078	-0.154	-0.241	0.024	0.006
eigen-worms	-0.048	0.059	-0.019	-0.263	**0.323**	-0.074	0.017	-0.276
hydraulic	-0.010	0.104	-0.034	-0.166	0.019	-0.065	**0.272**	-0.232
lorenz-attractor	0.037	0.110	0.003	-0.287	-0.138	**0.162**	0.004	-0.190
mocap	0.025	0.047	-0.071	0.174	0.096	0.050	nan	**0.225**
occupancy	-0.113	**0.214**	-0.058	-0.039	-0.258	-0.285	0.151	-0.203
own-synth	-0.039	0.159	-0.033	nan	0.028	**0.279**	0.272	-0.346
thomas-attractor	-0.036	nan	-0.018	nan	**0.232**	0.053	-0.045	-0.217

TABLE 12. Metric 'metrics.calinski-harabasz' value for each dataset and algorithm from default calibration results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	39.626	**169.623**	25.065	139.912	nan	113.178	43.804	5.369
cmapss	184.909	**3.84e+04**	567.915	2635.822	248.424	1798.528	599.298	871.293
eigen-worms	353.541	**2498.356**	74.512	327.072	1.306	876.894	117.145	55.120
hydraulic	210.616	**4612.781**	112.687	1574.895	63.695	631.183	5.678	4.147
lorenz-attractor	234.222	17.658	48.203	80.547	320.720	**1060.803**	190.984	18.462
mocap	64.989	2179.353	338.748	1954.337	1219.938	707.485	nan	**6561.213**
occupancy	352.260	**9190.251**	78.710	2738.856	879.927	3033.238	1099.447	171.655
own-synth	22.215	74.232	22.415	nan	42.333	1331.381	**2468.620**	21.841
thomas-attractor	64.071	nan	37.748	nan	0.387	**1319.776**	97.986	101.978

on a three-dimensional synthetic data set. The input data consists of four different operation points with small white noise. The sequence of the four subsequences is repeated once. The other rows are described in the caption of Figure 9. It is evident that the algorithm needs a few time steps to adapt to the current subsequence until it is recognized as such. Recognizing a previously identified subsequence, however, is almost instantaneous. The calibration of the thresholds η (horizontal black line) and ζ (horizontal gray line) are apparently crucial. The necessary time steps to adapt to a current subsequence can be altered by the calibration of the learning rate α and the number of BAE's training cycles

TABLE 13. Metric 'metrics.davies-bouldin' value for each dataset and algorithm from default calibration results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	30.023	1.424	11.749	4.098	nan	4.646	3.203	**0.593**
cmapss	3.926	**0.469**	16.567	1.017	3.670	1.483	1.798	2.143
eigen-worms	4.650	0.866	20.563	2.191	**0.859**	1.878	5.871	1.481
hydraulic	5.295	2.947	21.356	5.834	**1.645**	3.233	3.030	4.407
lorenz-attractor	4.899	**0.981**	9.134	9.212	3.306	1.334	3.412	1.674
mocap	7.686	**0.173**	1.686	0.531	0.866	1.447	nan	0.816
occupancy	4.536	5.129	35.854	2.850	2.901	**1.534**	1.609	3.231
own-synth	8.585	8.29e+04	14.357	nan	2.781	1.258	**0.759**	1.684
thomas-attractor	10.772	nan	14.992	nan	1.536	1.303	6.779	**0.874**

TABLE 14. Metric 'metrics.silhouette' value for each dataset and algorithm from default calibration results.

algorithm dataset	birch	bocpdetector	clustream	dbstream	denstream	mini-batch-kmeans	streamkmeans	abimca
bee-waggle	0.012	-0.104	-0.072	-0.098	nan	0.015	0.061	**0.216**
cmapss	0.139	0.007	-0.280	**0.494**	-0.168	-0.084	0.319	0.357
eigen-worms	0.082	0.025	-0.046	0.018	0.016	**0.112**	0.042	-0.165
hydraulic	0.128	-0.272	-0.113	-0.465	-0.207	0.007	**0.264**	-0.525
lorenz-attractor	0.112	-0.117	-0.064	-0.099	-0.169	**0.211**	0.117	-0.416
mocap	0.031	0.020	0.008	0.209	**0.416**	0.231	nan	0.375
occupancy	0.094	-0.424	-0.106	-0.053	-0.438	-0.110	**0.126**	-0.301
own-synth	0.012	-0.233	-0.051	nan	-0.475	0.387	**0.505**	-0.505
thomas-attractor	-0.003	nan	-0.045	nan	-0.177	**0.192**	0.019	-0.290

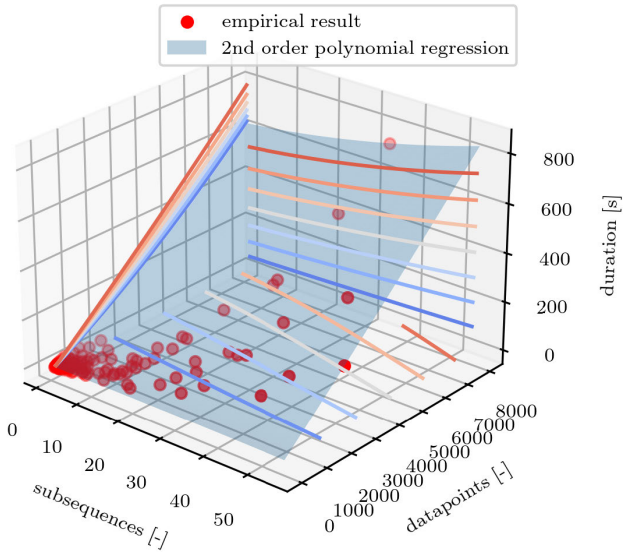

FIGURE 12. Empricial complexity estimation with variation of total number of datapoints and subsequences identified by our algorithm over the duration. 11.

per time step ω. A faster recognition of a subsequence has the drawback of the algorithm being very sensitive and therefore identifying even small changes of the input as a new subsequence. A strategy could be to calibrate the algorithm first to be rather insensitive and cluster the time-series in major subsequences. These can then be further clustered with a more sensitive calibration. This procedure can be repeated until the required degree of granularity is achieved. For a streaming application multiple runs of this procedure could also be applied in parallel and combined into a cluster tree.

B. EVALUATION

For the evaluation study of our algorithm, we chose eight different MTS datasets (see Table 3) from which six are publicly available and two are provided with our codebase [68], seven other state-of-the-art algorithms (see Table 1) and three widely used unsupervised clustering metrics (see Table 2). Each algorithm has been applied to each dataset with default parameters. Additionally, we performed a hyperparameter search for each algorithm based on a random grid search of 300 samples. The parameter boundaries for this hyperparameter search are listed in Table 15. Overall, 19 264 experiments were run.

For a better overview of the results, we chose to compare every algorithm to the "MiniBatchKMeans" algorithm and counted the number of times they performed better. Table 5 shows the results for the hyperparameter search and the number of outperformances of each algorithm compared to the "MiniBatchKMeans" algorithm. Table 6 shows the same results with default parameter settings for each algorithm. We can see, that in sum and in two of the metrics our algorithm beats state-of-the-art algorithms. The full list of results is attached in Table 7 and 14. Additionally, Table 16 shows the best results from the hyperparameter search when sorted by *MT3SCM* with the total time spent.[11]

[11]Experiments were performed on a Linux machine with a AMD Ryzen Threadripper 2950X 16-Core Processor using a GeForce RTX 2080 GPU.

Algorithm 2 ABIMCA. Using the Following Parameters: α: Learning Rate, ω: Number of Base Model Training Cycle per Step, η: Subsequence Detection Score Threshold, ρ: Subsequence Recognition Score Threshold, Window Length: ζ. Also, We Denote θ_S as a List of Subsequence Model Parameters, $\boldsymbol{s}$ as an Array of Scores for All Existing Subsequence Models, $\widetilde{W}_t$ the Reconstruction of the Sliding Window Input,

procedure Abimca(W_t) ▷ Sliding window data $W_t \in \mathbb{R}^{d\times\zeta}$
  Verify calibration $t \geq \zeta \vee \alpha > 0 \vee \omega \geq 1 \vee \eta > 0 \vee \rho > \eta$
  $c_S \leftarrow 0$ ▷ Initialize subsequence counter
  $\theta_b \leftarrow$ Sparse($sparsity = 0.1$) ▷ Initialize base model parameters
  while W_t **do** ▷ New input available
    for $j \leftarrow 1, \omega$ **do** ▷ Iterative base model train loop
      $\widetilde{W}_t, \mathbf{h}_b \leftarrow$Predict($W_t, \theta_b$)
      $l_b \leftarrow \mathcal{L}(W_t, \widetilde{W}_t, \mathbf{h})$
      $\Delta\theta_b \leftarrow$Backpropagate($l_b$)
      $\theta_b \leftarrow \theta_b + \Delta\theta_b$ ▷ Update base model parameters
    end for
    $\widetilde{W}_t, \mathbf{h}_b \leftarrow$Predict($W_t, \theta_b$)
    $s_b \leftarrow$SF($l_b, \mathbf{h}_b$)
    $\boldsymbol{s} \leftarrow$GetSubsequenceScores($W_t, \theta_\mathbf{S}$)
    if min($\boldsymbol{s}$) $< \rho$ **then** ▷ Subsequence recognized
      $S_{ID} \leftarrow \arg\min(\boldsymbol{s})$ ▷ Set ID to recognized subsequence index
    else ▷ No subsequence recognized
      if $s_b <= \eta$ **then** ▷ Score below new subsequence threshold
        $\theta_\mathbf{S}.append(\theta_b)$ ▷ Append current base model parameters to list of subsequence models parameters
        $c_s \leftarrow c_s + 1$ ▷ increase subsequence counter
        $S_{ID} \leftarrow c_s$ ▷ Set ID to new subsequence index
      else ▷ In transition
        $S_{ID} \leftarrow 0$ ▷ Set ID to unknown
      end if
    end if
  end while
end procedure

To estimate the complexity of the algorithm additional experiments were run to substantiate our theoretical consideration for the Big-O notation (see Figure 12). Since it is an online algorithm that uses a sliding window, the duration of the algorithm is linearly correlated to the number of data points to be analyzed. Additionally, with every subsequence

TABLE 15. Hyperparameter random grid search upper and lower bound for each algorithm and their specific parameter options.

algorithm	parameter	bound	value
birch	threshold	lower	0.000
	branching-factor	lower	2.000
	n-clusters	lower	2.000
	seq-len	lower	1.000
	threshold	upper	1.000
	branching-factor	upper	100.000
	n-clusters	upper	20.000
	seq-len	upper	30.000
streamkmeans	chunk-size	lower	1.000
	n-clusters	lower	2.000
	chunk-size	upper	50.000
	n-clusters	upper	50.000
abimca	learning-rate	lower	0.000
	omega	lower	5.000
	step-size	lower	1.000
	seq-len	lower	5.000
	eta	lower	0.001
	theta-factor	lower	1.000
	learning-rate	upper	0.010
	omega	upper	15.000
	step-size	upper	3.000
	seq-len	upper	20.000
	eta	upper	10.000
	theta-factor	upper	3.000
clustream	time-window	lower	1.000
	max-micro-clusters	lower	1.000
	n-macro-clusters	lower	1.000
	micro-cluster-r-factor	lower	1.000
	time-window	upper	20.000
	max-micro-clusters	upper	10.000
	n-macro-clusters	upper	50.000
	micro-cluster-r-factor	upper	4.000
dbstream	clustering-threshold	lower	0.100
	fading-factor	lower	0.001
	cleanup-interval	lower	1.000
	minimum-weight	lower	0.100
	intersection-factor	lower	0.100
	clustering-threshold	upper	5.000
	fading-factor	upper	10.000
	cleanup-interval	upper	10.000
	minimum-weight	upper	5.000
	intersection-factor	upper	1.000
mini-batch-kmeans	n-clusters	lower	1.000
	max-iter	lower	1.000
	batch-size	lower	128.000
	seq-len	lower	1.000
	n-clusters	upper	30.000
	max-iter	upper	200.000
	batch-size	upper	2048.000
	seq-len	upper	100.000
denstream	decaying-factor	lower	0.000
	beta	lower	0.000
	mu	lower	0.000
	epsilon	lower	0.000
	n-samples-init	lower	1.000
	stream-speed	lower	1.000
	decaying-factor	upper	1.000
	beta	upper	5.000
	mu	upper	5.000
	epsilon	upper	1.000
	n-samples-init	upper	1000.000
	stream-speed	upper	1000.000
bocpdetector	lag	lower	1.000
	changepoint-prior	lower	0.001
	threshold	lower	0.000
	lag	upper	50.000
	changepoint-prior	upper	10.000
	threshold	upper	5.000

TABLE 16. Metrics from hyperparameter search when sorted for best value of mt3scm metric.

algorithm	dataset	mt3scm	mt3scm-wcc	silhouette	calinski-harabasz	davies-bouldin	time-total-s
birch	bee-waggle	0.139	0.051	**0.181**	21.861	4.003	0.706
	cmapss	0.129	0.183	**0.461**	3918.728	1.485	4.922
	eigen-worms	0.018	0.002	0.104	530.308	2.999	7.885
	hydraulic	0.364	0.095	0.588	647.983	0.511	5.112
	lorenz-attractor	0.242	0.079	0.380	876.824	1.057	2.764
	mocap	0.102	0.238	0.052	41.244	5.665	26.209
	occupancy	0.430	0.085	0.201	155.363	3.749	4.662
	own-synth	0.355	0.382	0.287	1974.332	0.785	1.042
	thomas-attractor	0.115	**0.064**	**0.295**	**1953.123**	1.097	4.385
bocpdetector	bee-waggle	0.143	**0.430**	-0.104	**169.623**	**1.424**	2.373
	cmapss	0.120	**0.360**	0.007	**3.84e+04**	**0.469**	173.916
	eigen-worms	0.059	**0.178**	0.025	**2498.356**	**0.866**	89.426
	hydraulic	0.104	**0.312**	-0.272	**4612.781**	2.947	15.520
	lorenz-attractor	0.110	0.330	-0.117	17.658	0.981	19.703
	mocap	0.047	0.140	0.020	2179.353	**0.173**	106.425
	occupancy	0.214	**0.642**	-0.424	9190.251	5.129	74.403
	own-synth	0.159	0.478	-0.233	74.232	8.29e+04	5.606
clustream	bee-waggle	0.125	0.335	-0.105	1.843	10.309	0.192
	cmapss	0.213	0.309	0.375	1951.437	1.552	5.513
	eigen-worms	0.099	0.016	-0.029	16.995	8.453	2.552
	hydraulic	0.055	0.011	-0.198	5.403	6.311	0.838
	lorenz-attractor	0.019	0.064	-0.057	8.462	12.831	1.140
	mocap	0.256	0.409	-0.049	118.781	1.565	41.783
	occupancy	0.084	0.128	-0.219	170.070	2.603	2.851
	own-synth	0.096	0.095	-0.061	0.254	43.368	0.292
	thomas-attractor	0.013	0.013	-0.079	3.639	23.870	1.584
dbstream	bee-waggle	0.051	0.070	0.092	92.819	2.195	0.081
	cmapss	0.108	0.341	0.187	540.812	0.981	6.540
	eigen-worms	-0.026	0.001	-0.053	236.742	2.490	1.019
	hydraulic	0.006	0.037	-0.034	1098.636	1.608	35.551
	lorenz-attractor	0.272	0.170	0.014	103.571	6.533	0.324
	mocap	**0.277**	**0.832**	**0.357**	**7.28e+04**	0.479	59.624
	occupancy	**0.697**	0.096	**0.774**	45.932	**0.216**	**0.595**
	own-synth	0.366	**0.866**	**0.439**	**2.29e+04**	**0.598**	3.134
denstream	bee-waggle	0.183	0.046	-0.265	7.360	1.925	0.919
	cmapss	0.226	0.264	-0.037	769.884	5.774	28.740
	eigen-worms	**0.389**	0.014	0.028	230.673	1.056	1.282
	hydraulic	0.590	0.032	0.420	4.682	0.364	4.246
	lorenz-attractor	0.274	**0.349**	0.213	669.551	**0.930**	12.230
	mocap	0.273	0.469	-0.140	132.983	1.097	1.224
	occupancy	0.450	0.109	0.766	**1.63e+04**	0.366	0.787
	own-synth	0.279	0.140	0.279	778.017	1.821	0.567
	thomas-attractor	0.151	0.001	0.269	88.523	**0.587**	1.708
mini-batch-kmeans	bee-waggle	0.075	0.247	0.114	109.755	3.987	0.407
	cmapss	0.141	0.170	-0.053	4240.889	2.769	2.625
	eigen-worms	0.064	-0.001	**0.181**	1008.417	1.391	3.280
	hydraulic	0.655	0.182	**0.761**	3307.874	**0.338**	0.714
	lorenz-attractor	0.279	0.161	0.224	1223.689	1.244	1.736
	mocap	0.257	0.314	0.206	352.696	1.498	**0.972**
	occupancy	0.267	0.060	0.641	6790.810	0.658	2.954
	own-synth	0.366	0.325	0.396	1653.130	0.781	0.844
	thomas-attractor	0.153	0.011	0.201	1261.001	1.403	3.021
streamkmeans	bee-waggle	0.099	0.068	0.106	50.678	2.876	**0.063**
	cmapss	**0.378**	0.102	-0.337	411.260	1.891	**0.831**
	eigen-worms	0.099	0.003	0.130	271.766	1.894	**0.784**
	hydraulic	**0.667**	0.195	0.625	1791.171	1.034	**0.180**
	lorenz-attractor	0.272	0.159	0.366	**1958.559**	0.946	**0.240**
	mocap	0.067	0.095	0.115	143.753	2.005	7.911
	occupancy	0.458	0.224	0.531	5850.976	0.825	0.713
	own-synth	0.297	0.109	0.252	1030.563	1.772	**0.143**
	thomas-attractor	0.079	0.005	0.099	751.781	2.225	**0.478**
abimca	bee-waggle	**0.370**	0.039	-0.304	0.426	1.939	28.309
	cmapss	0.179	0.233	0.164	1717.623	3.207	246.458
	eigen-worms	0.150	0.004	0.151	834.686	1.585	212.060
	hydraulic	0.598	0.032	-0.385	0.882	1.191	67.832
	lorenz-attractor	**0.368**	0.073	**0.432**	1499.546	0.973	223.865
	mocap	0.258	0.288	-0.070	9.057	0.896	31.562
	occupancy	0.235	0.018	-0.005	100.938	2.681	102.439
	own-synth	**0.622**	0.080	0.128	120.002	0.648	117.261
	thomas-attractor	**0.474**	0.002	0.109	3.655	0.711	123.709

identified, the algorithm checks if the new incoming data is already known by comparing it to the previously identified subsequences. A linear correlation of the duration for every datapoint with the number of subsequences identified is therefore present. The complexity of our algorithm is therefore $O(NS)$, where N is the number of datapoints and S is the number of subsequences identified. Considering the worst case scenario of identifying every new datapoint as a new subsequence the duration for the algorithm increases quadratically with the number of datapoints, which results in a complexity of $O(n^2)$.

As cited before, every algorithm performs differently on the specific distribution of patterns and the hyperparameter search was a simple random grid search of ''only'' 300 samples, so these results unlikely represent the optimal solution for each algorithm on each dataset. Nevertheless, we demonstrate, that the algorithm we present in this work, is highly effective of detecting subsequences online in a MTS.

VII. LIMITATIONS AND DISCUSSION

For the evaluation of the segmentation of MTSD, we introduced a new metric which is based on space-curve parameters in the feature space. Due to the wide variety of fields, use cases and applications, this falls into place for some applications and uses cases but not for all. The calculation of these space-curve parameters is sensitive to outliers and smoothness and questionable for steady-state conditions or *non-moving* point clouds in the feature space. We have implemented specific numerical boundary limits for computing the derivatives of the data in these states, but it needs to be considered and evaluated if this is compatible with the application. Because of the outlier sensitivity we use the mean value of these parameters as well as their standard deviation. A low-pass filter for very noisy data should be considered before applying it to the metric. Attention is called for, when the data is scaled or standardized. This effects the actual space curve parameters, since a constant curvature is likely not constant anymore after scaling. The metric also tends to reward short subsequences who only occur once. Due to the mean value of the curve parameters the subsequence separation appears to be good, but the variance of one large subsequence is high. This needs to be compensated or prevented more and will be part of future analysis and improvement of the metric. It might also be reasonable to introduce weighting factors for the three parts of our metric in Equation (12) to consider domain specific emphasis on a rather spatial or curve parameter separation requirement.

Regarding our clustering method, calibration of the main thresholds (η, ρ) needs special attention. In combination with the learning rate, they mainly influence the ''sensitivity'' of the segmentation process. Using only the reconstruction error or only the representation deviation can be beneficial for different use cases and complexities (e.g., changing the weighting factor in Equation (34)). Advantages are that no additional information about the data for offline clustering needs to be saved. All necessary information for clustering data after training is the parameters of the subsequence specific AEs. It is a completely unsupervised method which can cluster online data. In the context of CbM the once identified subsequence AEs can be used for deviation quantification of the underlying system. This can be used for deterioration analysis and maintenance strategies. Further investigations for improving the ABIMCA method would be to explore different kind of AEs like feedforward neural network (FNN), CNN or a combination of such. Also, a VAE could be reasonable depending on the underlying process. Future work should analyze the effect of reducing the latent space dimension by multiple factors of the input dimension (when input dimension is very high). This could reduce computation costs and improve representation learning without performance loss. A detailed analysis of the optimal default parameters or a generic automatic calibration depending on some statistics of the expected input could increase performance and decrease calibration efforts.

VIII. CONCLUSION

In this paper we have introduced the Autoencoder-based Iterative Modeling and Subsequence Clustering Algorithm (ABIMCA) which is a deep learning method to separate multivariate time-series data (MTSD) into subsequences. It is beneficial in a variety of fields, to cluster MTSD into smaller segments or subsequences in an unsupervised manner. The ability to filter measurement data based on specific subsequences can improve downstream development products such as anomaly detection or machine diagnosis in Condition-based Maintenance (CbM) strategies. Our algorithm is specifically useful for MTSD generated by a mechatronic system in a transient environment. It can be used offline as well as online for streaming data. It utilizes recurrent neural network (RNN) based Autoencoders (AE) by iteratively training a Base Autoencoder (BAE), generating a segmentation score and saving the intermediate parameters of the BAE to recognize previously identified subsequences. By comparing our algorithm with seven other algorithms on eight different publicly available datasets using four different unsupervised metrics (from which we introduced one ourselves), we have shown that our algorithm outperforms state-of-the-art algorithms. Our unsupervised metric introduced (Multivariate Time-Series Sub-Sequence Clustering Metric (*MT3SCM*)), is an attempt to use a more intuitive similarity measure based on the curvature and other space-curve parameters of the spanned feature space. Additionally, all our code is open source and publicly available for benchmarking.

REFERENCES

[1] E. Quatrini, F. Costantino, G. Di Gravio, and R. Patriarca, ''Condition-based maintenance—An extensive literature review,'' *Machines*, vol. 8, no. 2, p. 31, Jun. 2020.
[2] R. Isermann, *Fault-Diagnosis Systems: An Introduction From Fault Detection to Fault Tolerance*. Berlin, Germany: Springer, 2006.
[3] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control* (Wiley series in Probability and Statistics), 5th ed. Hoboken, NJ, USA: Wiley, 2016. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1061322

[4] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information science and statistics), 1st ed. New York, NY, USA: Springer, 2016.

[5] J. F. Roddick and M. Spiliopoulou, "A survey of temporal knowledge discovery paradigms and methods," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 4, pp. 750–767, Jul. 2002.

[6] J. Lin, E. Keogh, S. Lonardi, and P. Patel, "Finding motifs in time series," *Proc. 2nd Workshop Temporal Data Mining*, 2002, pp. 1–11.

[7] G. J. J. van den Burg and C. K. I. Williams, "An evaluation of change point detection algorithms," 2020, *arXiv:2003.06222*.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[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] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. 11th Int. Conf. Data Eng.*, Mar. 1995, pp. 3–14.

[11] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, 2015.

[12] M. Lovrić, M. Milanović, and M. Stamenković, "Algoritmic methods for segmentation of time series: An overview," *J. Contemp. Econ. Bus. Issues*, vol. 1, no. 1, pp. 31–53, 2014. [Online]. Available: http://hdl.handle.net/10419/147468

[13] S. Torkamani and V. Lohweg, "Survey on time series motif discovery," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 7, no. 2, p. e1199, Mar. 2017.

[14] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 339–367, 2017.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, 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, pp. 2825–2830, Oct. 2011.

[16] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, vol. 25, no. 2, 1996, pp. 103–114.

[17] R. Prescott Adams and D. J. C. MacKay, "Bayesian online changepoint detection," 2007, *arXiv:0710.3742*.

[18] J. Montiel, M. Halford, S. Martiello Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. Murilo Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: Machine learning for streaming data in Python," 2020, *arXiv:2012.04740*.

[19] C. C. Aggarwal, P. S. Yu, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proc. VLDB Conf.* Elsevier, 2003, pp. 81–92.

[20] M. Hahsler and M. Bolaños, "Clustering data streams based on shared density between micro-clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1449–1461, Jun. 2016.

[21] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SIAM Int. Conf. Data Mining*, J. Ghosh, D. Lambert, D. Skillicorn, and J. Srivastava, Eds. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006, pp. 328–339.

[22] D. Sculley, "Web-scale K-means clustering," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, New York, NY, USA, M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, Eds. 2010, p. 1177.

[23] T. W. Liao, "Clustering of time series data—A survey," *Pattern Recognit.*, vol. 38, no. 11, pp. 1857–1874, 2005.

[24] S. Zolhavarieh, S. Aghabozorgi, and Y. W. Teh, "A review of subsequence time series clustering," *Sci. World J.*, vol. 2014, Jul. 2014, Art. no. 312521.

[25] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering—A decade review," *Inf. Syst.*, vol. 53, pp. 16–38, Oct. 2015.

[26] M. Carnein and H. Trautmann, "Optimizing data stream representation: An extensive survey on stream clustering algorithms," *Bus. Inf. Syst. Eng.*, vol. 61, no. 3, pp. 277–297, Jun. 2019.

[27] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data," *ACM Trans. Knowl. Discovery Data*, vol. 3, no. 1, pp. 1–58, Mar. 2009.

[28] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Process.*, vol. 167, Feb. 2020, Art. no. 107299.

[29] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," 2018, *arXiv:1801.07648*.

[30] C. Isaksson, M. H. Dunham, and M. Hahsler, "SOStream: Self organizing density-based clustering over data stream," in *Machine Learning and Data Mining in Pattern Recognition* (Lecture Notes in Computer Science), vol. 7376, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. P. Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and P. Perner, Eds. Berlin, Germany: Springer, 2012, pp. 264–278.

[31] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231. [Online]. Available: http://dl.acm.org/citation.cfm?id=3001460.3001507

[32] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 685–694.

[33] M. Ghesmoune, M. Lebbah, and H. Azzag, "State-of-the-art on clustering data streams," *Big Data Analytics*, vol. 1, no. 1, Dec. 2016.

[34] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," in *Data Mining in Time Series Databases*, vol. 57, M. Last, A. Kandel, and H. Bunke, Eds. Singapore: World Scientific, 2003, pp. 1–21.

[35] M. Ramoni, P. Sebastiani, and P. Cohen, "Bayesian clustering by dynamics," *Mach. Learn.*, vol. 47, no. 1, pp. 91–121, 2002.

[36] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," *Data Mining Knowl. Discovery*, vol. 13, no. 3, pp. 335–364, 2006.

[37] R. P. Silva, B. B. Zarpelão, A. Cano, and S. B. Junior, "Time series segmentation based on stationarity analysis to improve new samples prediction," *Sensors*, vol. 21, no. 21, p. 7333, Nov. 2021.

[38] S. Lu and S. Huang, "Segmentation of multivariate industrial time series data based on dynamic latent variable predictability," *IEEE Access*, vol. 8, pp. 112092–112103, 2020.

[39] M. Ceci, R. Corizzo, N. Japkowicz, P. Mignone, and G. Pio, "ECHAD: Embedding-based change detection from multivariate time series in smart grids," *IEEE Access*, vol. 8, pp. 156053–156066, 2020.

[40] R. Li, S. Li, K. Xu, X. Li, J. Lu, and M. Zeng, "A novel symmetric stacked autoencoder for adversarial domain adaptation under variable speed," *IEEE Access*, vol. 10, pp. 24678–24689, 2022.

[41] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Spatio-temporal convolutional sparse auto-encoder for sequence classification," in *Proc. Brit. Mach. Vis. Conf.*, R. Bowden, J. Collomosse, and K. Mikolajczyk, Eds. 2012, p. 124.

[42] K. Bascol, R. Emonet, E. Fromont, and J.-M. Odobez, "Unsupervised interpretable pattern discovery in time series using autoencoders," in *Structural, Syntactic, and Statistical Pattern Recognition* (Lecture Notes in Computer Science), vol. 10029, A. Robles-Kelly, M. Loog, B. Biggio, F. Escolano, and R. Wilson, Eds. Cham, Switzerland: Springer, 2016, pp. 427–438.

[43] S. E. Chazan, S. Gannot, and J. Goldberger, "Deep clustering based on a mixture of autoencoders," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Oct. 2019, pp. 1–6.

[44] B. Yang, X. Fu, D. N. Sidiropoulos, and M. Hong, "Towards K-means-friendly spaces: Simultaneous deep learning and clustering," in *Proc. 34th Int. Conf. Mach. Learn.*, D. Precup and Y. W. Teh, Eds. vol. 70, 2017, pp. 3861–3870. [Online]. Available: https://proceedings.mlr.press/v70/yang17b.html

[45] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li, "Multidimensional time series anomaly detection: A gru-based Gaussian mixture variational autoencoder approach," in *Proc. 10th Asian Conf. Mach. Learn.*, J. Zhu and I. Takeuchi, Eds. vol. 95, 2018, pp. 97–112. [Online]. Available: http://proceedings.mlr.press/v95/guo18a.html

[46] T. Chen, X. Liu, B. Xia, W. Wang, and Y. Lai, "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder," *IEEE Access*, vol. 8, pp. 47072–47081, 2020.

[47] W.-H. Lee, J. Ortiz, B. Ko, and R. Lee, "Time series segmentation through automatic feature learning," 2018, *arXiv:1801.05394*.

[48] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[49] (Sep. 14, 2020). *Data.gov*. [Online]. Available: https://data.gov/

[50] C. J. van Rijsbergen, *Information Retrieval*. London, U.K.: Butterworths, 1979.

[51] H. Kremer, P. Kranen, T. Jansen, T. Seidl, A. Bifet, G. Holmes, and B. Pfahringer, "An effective evaluation measure for clustering on evolving data streams," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, C. Apte, J. Ghosh, and P. Smyth, Eds. 2011, p. 868.
[52] J. Serrá and J. L. Arcos, "An empirical evaluation of similarity measures for time series classification," *Knowl.-Based Syst.*, vol. 67, pp. 305–314, Sep. 2014.
[53] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, pp. 53–65, 1987.
[54] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Stat., Theory Methods*, vol. 3, no. 1, pp. 1–27, 1974.
[55] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.
[56] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert, "Learning and inferring motion patterns using parametric segmental switching linear dynamic systems," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 103–124, May 2008.
[57] M. Arias Chao, C. Kulkarni, K. Goebel, and O. Fink, "Aircraft engine Run-to-Failure dataset under real flight conditions for prognostics and diagnostics," *Data*, vol. 6, no. 1, p. 5, Jan. 2021.
[58] E. Yemini, T. Jucikas, L. J. Grundy, A. E. X. Brown, and W. R. Schafer, "A database of caenorhabditis elegans behavioral phenotypes," *Nature Methods*, vol. 10, no. 9, pp. 877–879, Sep. 2013.
[59] T. Schneider, N. Helwig, and A. Schütze, "Automatic feature extraction and selection for classification of cyclical time series data," *Technisches Messen*, vol. 84, no. 3, pp. 198–206, Mar. 2017.
[60] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130–141, 1963.
[61] (Apr. 21, 2022). *Carnegie Mellon University—CMU Graphics Lab—Motion Capture Library*. [Online]. Available: http://mocap.cs.cmu.edu/
[62] L. M. I. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and Co_2 measurements using statistical learning models," *Energy Buildings*, vol. 112, pp. 28–39, Jan. 2016.
[63] R. Thomas, "Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis, 'labyrinth chaos,'" *Int. J. Bifurcation Chaos*, vol. 9, no. 10, pp. 1889–1905, 1999.
[64] J. Köhne. (2022). *MT3SCM: Multivariate Time Series Sub-Sequence Clustering Metric*. Python. [Online]. Available: https://github.com/Jokonu/mt3scm
[65] C. S. Möller-Levet, F. Klawonn, K.-H. Cho, and O. Wolkenhauer, "Fuzzy clustering of short time-series and unevenly distributed sampling points," in *Advances in Intelligent Data Analysis V* (Lecture Notes in Computer Science), vol. 2810, G. Goos, J. Hartmanis, J. van Leeuwen, M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, Eds. Berlin, Germany: Springer, 2003, pp. 330–340.
[66] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[67] M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proc. NIPS*, 2006, pp. 1–8.
[68] J. Köhne. (2022). *ABIMCA: Autoencoder Based Iterative Modeling and Subsequence Clustering Algorithm*. Python. [Online]. Available: https://github.com/Jokonu/abimca

JONAS KÖHNE was born in Berlin, Germany, in 1983. He received the Dipl.-Ing. degree in energy and process engineering from the Technische Universität Berlin, Germany, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Energy and Automation Technology. From 2014 to 2015, he worked as a Function Development Engineer and an Embedded Software Test Engineer in the automotive field of electrical power steering with Bertrandt AG. From 2015 to 2019, he worked as a Function Developer of powertrain and power engineering with the Department of Commercial Vehicle Electronics, IAV GmbH. Since 2019, he has been a Research Assistant with the Department of Energy and Automation Technology, Technische Universität Berlin. He is also working as a Data Scientist with the Department Commercial Vehicle Electronics, IAV GmbH. His research interests include anomaly and novelty detection, condition/predictive maintenance strategies of mechatronic systems, analysis of multivariate time-series data using autoencoder, and fault discovery and identification.

LARS HENNING was born in Nauen, Germany, in 1975. He received the Dipl.-Ing. degree in energy and process engineering from the Technische Universität Berlin, Germany, in 2002, and the Ph.D. degree in control engineering, in 2008. Since 2008, he has been working with the Department of Commercial Vehicle Electronics, IAV GmbH, in the area of powertrain and power engineering. Currently, he is a Team Manager of the Software Development Team, with the focus on condition/predictive maintenance strategies of mechatronic systems and machine-learned algorithms for powertrain control.

CLEMENS GÜHMANN was born in Berlin, Germany, in 1962. He received the Dipl.-Ing. degree in electrical engineering from the Technische Universität Berlin, Germany, in 1989, and the Ph.D. degree in pattern recognition and technical diagnosis, in 1995. From 1989 to 1994, he was a Research Assistant with the Institute for General Electrical Engineering. From 1994 to 1995, he worked as a Function Development Engineer with Whirlpool Corporation (Bauknecht). From 1996 to 2003, he was employed as a Function Development Engineer with IAV GmbH, where he was the last Head of the Department of Transmission Systems. In 2001, he received a Teaching Assignment with the Technische Universität Berlin, where he was appointed to a professorship at the Chair of Electronic Measurement and Diagnostic Technology, in 2003. His research interests include the measurement technology and data processing and the diagnosis, predictive maintenance, modeling, simulation, and automatic control of mechatronic systems.