IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 March 2024, accepted 9 May 2024, date of publication 14 May 2024, date of current version 22 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3400970

RESEARCH ARTICLE

Anomaly Detection in Smart Industrial Machinery Through Hidden Markov Models and Autoencoders

RADU SOROSTINEAN, ZAHARIA BURGHELEA, AND ARPAD GELLERT
Computer Science and Electrical Engineering Department, Lucian Blaga University of Sibiu, 550025 Sibiu, Romania

Corresponding author: Arpad Gellert (arpad.gellert@ulbsibiu.ro)

This work was supported by Lucian Blaga University of Sibiu (Knowledge Transfer Center) and Hasso Plattner Foundation Research under Grant LBUS-HPI-ERG-2023-03.

ABSTRACT This study addresses the need to develop a sustainable manufacturing process in industrial factories, as the industry desires to remain competitive while it is challenged to adopt eco-friendly practices. A Machine Learning based software is proposed to deal with the environmental issues, aiming to facilitate the monitoring and analysis of industrial machinery, more exactly of CNC woodworking machines. The focus is on two aspects that determine the environmental impact: energy consumption and toxic emissions, which are used to determine the operating modes of the machines and to detect potential working anomalies. This software consists of a pipeline with two main components: the first one aims to categorize the operating modes of the used machines through time series clustering methods, such as Hidden Markov Models. The second component employs Hidden Markov Models again alongside deep learning based Autoencoders to identify huge deviations within the environmental data. For evaluation, a dataset was collected as a time series from a CNC woodworking machine and then the preprocessed data was further analyzed using the implemented software. The experiments have shown that for anomaly detection in machine operating modes, the Hidden Markov Model outperforms the Autoencoder and state-of-the-art models in terms of efficiency and robustness.

INDEX TERMS Anomaly detection, autoencoders, hidden Markov models, Industry 4.0, long short-term memory, working mode detection.

I. INTRODUCTION

Sustainable manufacturing has become one of the most important environmental problems and it is moving towards eco-friendly practices in its various subfields, notably also in the manufacturing sector via the Industry 4.0. Industrial factories often face challenges when it comes to implementing eco-friendly practices due to limited knowledge and resources in production. Therefore, identifying and developing efficient methods to analyze and improve the manufacturing process can ultimately lead to minimizing the toxic environmental impact.

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

This study concentrates on novel approaches of integrating established methods to detect anomalies in industrial machinery. While we are fortunate enough to have a plethora of methods that deal with all kinds of anomaly detection, one of the biggest challenges is that many models struggle with the increasing dimensionality of the data. Although many models emerged that could handle many features theoretically, such as Vector Autoregressive models or Deep Learning based methods, e.g., Long Short-Term Memory (LSTM), the nature of the dimensionality may still provide an impediment in finding important correlation within the data. Some dimensionality reduction methods can be employed before the actual anomaly detection process, such as principal component analysis (PCA), averaging similar features or other space-embedding strategies [1].

Another challenge, especially in the case where the data is unlabeled, appears to be the evaluation metrics. Heuristics are necessary to flag possible outliers, such as the Pauta criterion (also known as 3-sigma rule) after pseudo-labeling the data.

Our study starts with a clustering process aimed to identify patterns within the time series, where the final goal is to categorize the operating modes of the CNC woodworking machines. Furthermore, those clusters are meant to aid the process of flagging anomalies to obtain valuable insights regarding the manufacturing process. Anomaly detection algorithms examine data streams, by extracting irregularities in the energy consumption patterns. These insights give manufacturers an advantage to identify inefficiencies in real-time, thereby reducing energy usage, lowering operational costs, and empowering eco-friendly production practices. Moreover, proactively identifying anomalies in machinery and equipment for manufacturers prevents potential breakdowns and malfunctions. This shifts the pace of maintenance from reactive to predictive, improving equipment longevity and reducing waste. Another advantage is resource optimization, deviations in resource usage, such as raw materials or water, can be detected and data driven decisions can be made in real time. Another operational aspect that can be achieved is worker safety. Abnormalities in the manufacturing process that can endanger the workforce can be detected and addressed. This proactive stance in improving worker safety not only averts costly incidents but also fosters a safe work environment.

The proposed method has two main components: the first component classifies the operating modes of the working machines using Hidden Markov Models (HMM), while the second focuses on flagging huge deviations within the environmental data, using yet again HMM, but also employing more modern techniques, such as deep learning based autoencoders (AE). These two components build a close connection, as the automatic clustering part is applied first to enable an easier flow in detecting anomalies. The novelty of the paper consists in the combination of the two components. By applying these techniques, this study aims to enable more sustainability in the manufacturing sectors. While some aspects of the software are particular for this specific woodworking industry, the core idea of fusing clustering and anomaly detection algorithms can be further applied to promote eco-friendliness in other fields as well.

The paper has the following structure: Section II reviews the relevant work in the field, and Section III outlines the theoretical aspects regarding the used algorithms. This is followed by Section IV, which presents the dataset and the preprocessing step and describes the implementation of the proposed working mode identification and anomaly detection methodology, with a focus on industrial machinery. Finally, Section V illustrates and discusses the experimental results, while Section VI concludes the paper by providing possible further work directions.

II. RELATED WORK

A. INDUSTRY 4.0

In the era of the fourth industrial revolution – often mentioned as Industry 4.0 [2] –, similarly to software versioning systems, manufacturing and production systems have undergone a transformative evolution. The advent of this new industrial era has given rise to numerous innovative concepts, each contributing to the realization of more sustainable and efficient manufacturing processes. Among these groundbreaking ideas, predictive maintenance has emerged as a central pillar, fundamentally altering the landscape of machine care and defining a digital age of maintenance practices [3]. As part of this revolution and adding layers of software on top of already existing infrastructure, a significant layer of cybersecurity threats is also exposed [4]. A systematic Industry 4.0 review [5] answers the question about what applications and business cases are enabled for companies by the Industry 4.0. Another industry review [6] explores the innovative realm of supply chain collaboration, particularly in the context of the circular economy.

This emerging field emphasizes close and novel forms of cooperation among supply chain stakeholders, extending beyond traditional industry boundaries, all with the goal of implementing sustainable circular systems. In [7], a smart environment is described as a place where the boundaries between the physical and virtual worlds blur, all orchestrated by intelligent cyber-physical systems. The very industrial environment that hosts such cutting-edge digital automation and information technology is an attractive target for cyberattacks. The focus of the study is using machine vision to detect anomalies or behaviors of machines caused by different cyber security threats. A digital twin is explored in [8] with the goal of detecting anomalies based on data from the machine learning algorithms, digital twin and Industry 4.0 technology. Anomaly detection is also explored in [9] by doing a review of the current techniques used in metal additive manufacturing.

B. TIME SERIES CLUSTERING

Clustering is a method which groups similar data into homogeneous categories without prior knowledge or information. In contrast, classification follows the same process, however with the groups being already defined. Specifically for this study, one type of clustering that will be performed is time series clustering, a time series being a sequence of values recorded at successive timesteps. The main difference over the classical clustering arises due to the fact that the data is manifesting temporal dependencies over time [10].

To elaborate further, the goal is to group the operating modes of the CNC woodworking machine. While the exact number of possible clusters is unknown, as we do not even know the specific machine used, we are aware that there must be some working modes (e.g., the machine can be shut down, cutting, standby and so on). Such modes might be observed for a longer period of time (e.g., if the machine is turned off overnight) or they can switch frequently

(e.g., in case the stand-by comes immediately after cutting one product and then this process is repeated).

One common algorithm used for time series clustering is K-Means, which falls into the category of Partitioning Clustering methods. K-Means divides the data into k distinct clusters by looking at the similarity between the data points and each cluster. While this is a simple, efficient and fast technique, the disadvantage lies in the fact that it inherently neglects the temporal dependencies [11].

On the opposite spectrum, there are Hierarchical Clustering methods, such as the Agglomerative Hierarchical Clustering (AHC) algorithm. In this case, each data point is considered a cluster initially and then the clusters are gradually merged until only the most relevant ones remain. As previously mentioned, the temporal information is not taken inherently into account in [10] and [11].

To overcome the previously mentioned limitation, other algorithms can be employed, such as the Model-Based Clustering methods. Within this category falls also the main algorithm used in this study, namely the HMM which does take into account the temporal dependencies. The idea of using HMM to categorize sequences of data can also be seen in a related field, namely applied to music genre classification as seen in [12]. While the music part is not directly relevant for our study, this highlights the application of a model-based algorithm on time series to automatically determine the genre of the melodies. Moreover, the study also shows a way on how to select the number of hidden states (or clusters) within the HMM, which is problematic as we are not aware of the number of possible operating modes within the working machines. A solution proposed in [12] is to increase the number of hidden states until no noticeable improvement is seen in performance.

An alternative for choosing the number of hidden states in a HMM-based application is described in [13], where a Monte-Carlo based cross-validation approach was proposed. Also, [14] goes in depth to discuss this notorious problem of determining the optimal number of hidden states and inclusively mentions the usage of Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). A particular case of those coefficients is further employed in our study.

C. ANOMALY DETECTION

Anomaly detection is the process of flagging data points which highly deviate from the expected pattern of the other items within the whole dataset. From detecting fraud transactions, diagnosing medical risks or identifying environmental problems, the usage of anomaly detection is widespread across many fields [15]. With regard to this project, anomaly detection is performed with the final goal of flagging huge deviations within the CNC woodworking machine data, which could indicate potential mechanical or environmental problems. Flagging such anomalies early can also prevent more serious issues down the line and make the process an eco-friendly one. Furthermore, a prior application of time series clustering can provide valuable information for the anomaly detection process, as by splitting the data into different working modes it can simplify the detection procedure for each operating mode, make the flow go smoother and give more reliability to the process.

One approach to identify anomalies, as described in [16], involves the application of joint statistical moments. By an analogy to PCA, the principal kurtosis vectors are used to signal the principal directions along which the outliers are most likely to appear. The kurtosis is serving as a reliable metric to detect anomalies, since it represents the ''tailedness'' of the probability distribution (or in other words, how often the outliers occur).

In [17], Hu et al. proposed a computational framework for anomaly detection in multivariate time series. First, a recurrence plot is generated, then abrupt changes are selected and, finally, anomalies are detected by computing the dissimilarities between any two subsequences. The proposed method proved to be efficient on a single dimensional dataset, two electrocardiogram datasets, a two-dimensional video surveillance dataset, and a five-dimensional dataset from a river tunnel.

HMMs, which were previously mentioned in the context of time series clustering, can also be adapted for anomaly detection as one application of the HMM is to compute the likelihood of observing a certain sequence given a model. By taking advantage of this, anomalies can be detected by identifying the sequences which are unlikely to belong to any of the hidden states [18].

Modern state-of-the-art methods for anomaly detection are often based on Deep Learning techniques, as described in [15]. One of them is the AE, which aims to learn a lower-dimensional representation space for the input and accurately reconstruct it. The same procedure can be found applied in data compression, where an AE can learn to encode and decode a particular signal or image. The core idea of using this technique in anomaly detection, is that the learned representation is enforced to be remembered in order to minimize the reconstruction error during the decoding part. Thus, since anomalies are much harder to be reconstructed, those decoded data points which significantly deviate from the original points are flagged as anomalies.

A special type of AE relies on a Recurrent Neural Network (RNN), specifically LSTM, and can be applied to identify anomalies in electrocardiography time signals, as described in [19]. The advantages of using LSTMs over classical RNNs are due the fact that they take into account temporal dependencies and are capable of remembering information over longer periods, making them more suitable for time series.

Furthermore, the authors in [20] employ an LSTM network to detect possible anomalies within the injection molding stage of the PET bottle production. This resembles this project, however the flow is reversed, as there the clustering

step is not used to aid the anomaly detection phase, but rather to further cluster the detected anomaly data in order to improve the process.

In [1], Ji et al. proposed a hybrid model based on space-embedding strategy (SES) to detect anomalies in multivariate time series. In the first stage, SES is used for dimensionality reduction and to determine the dissimilarities between adjacent subsequences. Then, the generated dissimilarity vector is processed by an LSTM model. Finally, a statistical method is used on the output of the LSTM to identify anomalies. Their experiments have shown high accuracy on four public datasets, but weaker performance on two real datasets.

Although the algorithms developed in this study are unsupervised and work with unlabeled data, a way to pseudo-label the data and measure the effectiveness of the algorithms and compare them was developed. The Pauta criterion [21], [22], [23], also referred to as the three-sigma rule, operates on the assumption that a dataset adheres, or approximately adheres, to a normal distribution and comprises solely random errors. This method involves calculating the standard deviation of the dataset and establishing an interval based on a specified probability. Errors falling outside this interval are identified as gross errors rather than random errors and are therefore flagged for elimination.

III. ANOMALY DETECTION IN MACHINE OPERATING MODES

A. HIDDEN MARKOV MODELS

A HMM is a stochastic model which represents systems directed in the background by hidden (or unobservable) states [24]. Even though those states are hidden, the emissions (or the observable outputs) are dependent on them and this leads to the possibility to analyze the hidden aspects, based on the observable outputs.

Each of the states possess a probability of transitioning to another state, which can also mean that it can remain in the same state (or in other words, perform a self-transition). Furthermore, these states are subject to a constraint in which the upcoming states depend solely on the current ones. As an example, if we were to model weather patterns, then a HMM assumes that if we desire to forecast tomorrow's weather, then we will do that based on today's weather, ignoring any information prior to this day. This principle is more concisely called the Markov Property, which is at the core of Markov chains [25].

Markov chains (or Markov processes) are memoryless stochastic processes that model the transition probabilities between sequences of random variables, each of those variables representing a state in the system. Formally, if we define the sequence of states $Q = q_i, i \in [1, n]$, then for any state k, the Markov Property can be described as:

$$P(q_k|q_{k-1}, \ldots, q_2, q_1) = P(q_k|q_{k-1}) \tag{1}$$

In addition to the state Q, a mathematical description of a Markov chain also includes a matrix holding the transition probabilities, $A = \{a_{ij}\}, i, j \in [1, n]$, where each a_{ij} represents the probability of switching from the state i to the state j. Initially, the system can be found according to a certain probability in any of the system's states, as given by the initial probability distribution: $P = \{p_i\}, i \in [1, n]$.

In contrast, the HMM is an extended variant of the Markov chain, augmented with an additional "layer" of emissions $B = \{b_j(o_k)\}$, each emission representing the likelihood of emitting the observation o_k while being in the q_j state. Above, $O = \{o_k\}, k \in [1, m]$ is the set of all possible observations. On top of the Markov Property, the nature of the model also brings out the Output Independence property:

$$P(o_k|q_1, \ldots, q_n, o_1, \ldots, o_m) = P(o_k|q_i) \tag{2}$$

That is, the probability of observing the output o_k is based only on the state that produced the output (in this case q_i) and not in any other state or observation [25].

In [26], Rabiner characterized HMMs by three fundamental problems: learning the model parameters, decoding the sequences and computing the likelihood of observing a specific sequence. In the context of this study, for a given time series clustering problem, learning the model parameters is essentially performing the training part. More exactly, the Baum-Welch algorithm, through an iterative process, learns the model parameters, such as the transition and the emission probabilities (the probability matrix corresponding to the transition between different clusters as well as the probability matrix representing the observations). After the model is trained, each timestamp can be assigned to one hidden state (or cluster) using the decoding algorithm (essentially this determines to which hidden state the instance most likely belongs to). This step is performed using the Viterbi Algorithm and takes advantage of dynamic programming to compute the most probable sequence of hidden states that led to the observed sequence (more exactly, features of the data are the observed part).

The time series clustering part can be enabled through the application of these two problems. Furthermore, the likelihood of observing a sequence of data can be employed to detect whether some potential anomalies occur within the time series. More exactly, when an instance has a low likelihood to belong to any of the hidden states, then that instance can be considered an outlier. This step is performed through the usage of the Forward Algorithm, which also employs dynamic programming.

B. THE LSTM MODEL

The modern anomaly detection is based upon deep learning techniques, starting from simple encoder-decoder RNNs. However, those pose a few limitations since only a few successive time steps are taken into consideration while training them. Also, problems such as the vanishing gradient are often encountered. Additionally, the layers associated with recurrent cycles are trying to make an accurate decision in the current timestep, while also trying to remember distant

information. Performing two tasks at once adds too much complexity to the process [25].

More advanced RNNs, such as the Long Short-Term Memory (LSTM) network, are addressing the previous limitations. These networks resemble (traditional) RNN with the difference that the recurrent cycle is replaced by a memory cell [27]. LSTM has already been studied in different industrial contexts [28], [29].

The name ''long short-term memory'' comes from the original paper [30] and reflects how the networks can use their recurrent cycles to memorize representations of recent input events in the form of outputs (seen as the short-term memory) for a long-term period embodied within the process of slowly changing weights.

As with standard RNNs, the data flowing into those memory cells (or LSTM cells), is the value of the hidden state (the output of the memory cell) from the previous timestep alongside the input from the current timestep. The process of how LSTMs work is detailed in [27] and further summarized below. LSTM cells (see Fig. 1) are formed out of an internal state (which is the memory that the cell holds through data processing) and three gates that control the flow of the information into, within and out of the cells:

- the input gate decides how much of an input should be stored in memory;
- the forget gate ponders whether some part of the information should be flushed or not;
- the output gate decides how much the memory information should impact the cell's output.

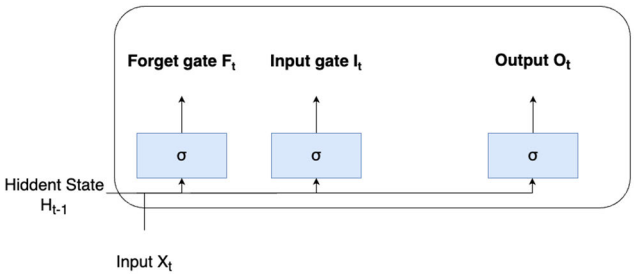


FIGURE 1. The architecture of a LSTM memory cell [27].

Essentially, the three gates are separate fully connected layers, with their outputs being dragged through an activation function – typically through the sigmoid function, denoted with σ, as its output is in the (0, 1) interval and this is ideal for creating a gate (or rather, it is easier for each gate to take a decision this way).

A single LSTM cell might not achieve that much on its own, but interconnected with other similar cells it builds up the structure of the layers and further that of the entire RNN.

What makes RNNs special is their ability to operate over sequences of vectors in multiple ways, as shown in Fig. 2. This includes structures such as: one-to-one (with fixed inputs and outputs, as for image classification); one-to-many (with a sequence vector output, as when the input is an image

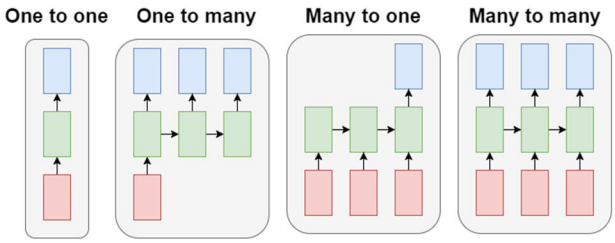


FIGURE 2. Different structures of RNNs [31].

and the output is a language description of it); many-to-one (the reverse of the one-to-many structure) and many-to-many (the most generalized variant, which is also the case in the context of this study, where both the input and the output are sequences of vectors) [31]. The flow within such a general structure goes as shown in Fig. 2. Specifically for a time series, the process starts with the first time step being the input for the first LSTM cell, then the second time step alongside the output of the first LSTM cell becomes the input for the second LSTM cell. The process goes further like this until the very end, but taking into account the different types of outputs (mentioned previously). Additionally, since the cells are encapsulated so that from an external perspective they will appear as basic units (similarly as with the concept of objects within object-oriented programming), it enables easily to experiment with different types of RNN architectures [25].

One such possible different architecture is the Bidirectional LSTM (Bi-LSTM), which essentially consists of two LSTMs: one that processes the data in the natural direction, whereas the other one reverses the direction, with both outputs from each network being concatenated at each timestep. While this is not feasible for online learning (where new data continuously appears), it is a major advantage for offline training (meaning that we are in possession of all the data), as there is simultaneous access to both the future and the past data from the network's perspective.

C. AUTOENCODERS

An AE is a special type of neural network which can efficiently compress its input data. The architecture consists of an Encoder (which aims to obtain a lower dimensional representation of the input data) and a Decoder (which attempts to reconstruct the original data based on the encoded representation) [32].

As an illustrative example, if we consider a simple feed-forward neural network with a single hidden layer and the size of the output layer is the same as for the input layer, then the network can be used as an AE by training it to minimize the error between the output and the original input itself, making an encoder-decoder process. More exactly, the output of the hidden layer is the resulting encoded representation. In case the hidden layer contains significantly fewer neurons than the input or the output layer, then this representation can be also viewed as the compressed data, assuming that

the programming language represents the resulting encoded representation using the same datatype as for the original input, which often might not be the case.

An LSTM-based AE is similar to the feedforward network exemplified in Fig. 3, with the difference that the architecture is built out of layers containing LSTM cells, as described in the previous section [33].

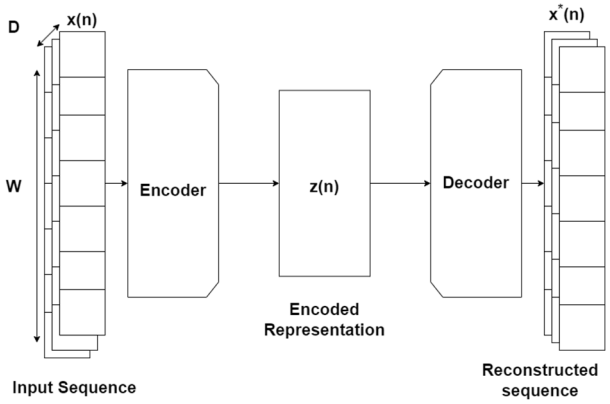


FIGURE 3. The architecture of an LSTM-based AE [33].

IV. EXPERIMENTAL METHODOLOGY

A. DATA PREPROCESSING

This project started by preprocessing the time series data, which was gathered in CSV files from a CNC woodworking machine. More exactly, the data consisted of two time series related to energy consumption and environment emissions. The exact type of the machine and its operating modes is, however, unknown.

Specifically, the energy data contained: the Active Power, the Current, the Power Factor, the Reactive Power, the Total Harmonic Current Distortion (THDI), the Total Harmonic Voltage Distortion (THDU) and the Voltage for each phase (A, B and C, as the machine was using a three-phase electric power system).

Meanwhile, the environmental data consisted of ambient conditions within the woodworking factory. These metrics encompass readings for ''sound'', ''pressure'', ''temp'', ''humidity'', and ''VOC'' used for environmental conditions. ''Organic chemicals'' and ''particles'' gauge the presence of these elements, while ''roll'', ''pitch'' and ''yaw'' quantify rotational movements about the X, Y, and Z axes, respectively. Accelerations along the X, Y, and Z axes are captured by ''Xacc'', ''Yacc'', and ''Zacc'', whereas ''Xgyr'', ''Ygyr'', and ''Zgyr'' correspondingly represent gyroscope values. Finally, metrics like ''pm1.0'', ''pm2.5'', and ''pm10'' delve into particulate matter of varying sizes, and ''CO2'' offers insight into carbon dioxide levels.

Given that the initial data had a frequency of 1 second, the time series were resampled by averaging them over 5 seconds. This was mainly performed in order to ignore the unnecessary high-frequency fluctuations and noise, thus focusing on longer patterns. After this step, any missing data was filled with the nearest non-null value within the data frame. Lastly, as the energy data came from a three-phase electric power system, each of the three phases were averaged in order to obtain a single representative value for each feature, thus, preserving the essential information while at the same time reducing the high-dimensionality of the data.

Before applying the HMM to cluster the time series, it was necessary to ensure that all features contribute equally to the model. This is performed by standardizing the data, which rescales the values so that the new mean is 0 and the new standard deviation is 1. This step is also essential to be applied before training the HMM algorithm, since the emissions are assumed to belong to a Gaussian distribution and having the values standardized can make the model more stable. The most appropriate model available for the used data was GaussianHMM which works with Gaussian emissions, as opposed to CategoricalHMM which assumes that the emissions are discrete.

B. DETERMINING THE OPTIMAL NUMBER OF CLUSTERS

The next step after standardizing the data was to determine the optimal number of hidden states, or in other words to find the number of clusters that the HMM algorithm should work with (a parameter that must be predefined). This can be found visually by looking at how HMM performs when changing the number of hidden states. However, in order to enable an automatic process, we desired to find the optimal number dynamically.

Some variants for determining the optimal number of states automatically were already employed by other authors, as described in Section II. Specifically, a common criterion that deals with this issue is the BIC, which is defined as:

$$BIC = k\ ln(n)\ - 2\ ln\ (\hat{L}) \tag{3}$$

Above, k is the number of parameters in the model, n is the data size (or the number of timesteps) and $\hat{L}$ is the maximized value of the likelihood of the model given the observed data (in simpler terms this likelihood is a measure of how accurately the model represents the data) [14].

However, for the data originating from the CNC machines, BIC had the unfortunate drawback of not penalizing the model enough, as it led to the situation where increasing the number of hidden states resulted in a decrease for the BIC value (and a smaller value of BIC means that the model performs better). In short, BIC always recommended choosing the largest possible number of states, but knowing how many operation modes the machine has, we could see that this should not be the case.

Alternative variants to BIC, such as applying AIC or performing cross-validation did not resolve this issue either (for the current dataset), as those also continued to suggest the maximum possible number of hidden states. To address this limitation, a modified version of BIC was further proposed in this study. More precisely, an Adjusted Bayesian Information Criterion (ABIC) was considered, which penalizes harder

models with a higher number of parameters. This criterion is further defined as:

$$ABIC = ln(\hat{L}) - k^2\, ln(n) \tag{4}$$

The main difference appears in the fact that the number of parameters (k) is squared in order to give a harder penalization. The minus sign was also reversed so that a higher value for ABIC now suggests that the model performs better (which felt more natural, as usually metrics with higher values tend to represent a better performance).

C. HMM IMPLEMENTATION

With the optimal number of hidden states determined, the next step was to cluster the time series. This was performed using the GaussianHMM model provided from the ''hmmlearn'' library.

The optimal number of clusters was determined using the ABIC criterion, and the covariance matrix of each component was chosen to be diagonal, meaning that the features are assumed to be uncorrelated within each component (hidden state or cluster). The training part is performed using the Baum-Welch algorithm within the ''fit'' method, whereas the ''predict'' function employs the Viterbi algorithm in order to decode to which component each observation belongs to.

Furthermore, the HMM can be employed to find potential anomalies within the predicted data. More exactly, the ''compute_log_likelihood'' function provides for each observation the log likelihood of belonging to any of the hidden states. Thus, by summing up the likelihoods of all the features within an observation and comparing it to a threshold can inform us about outliers.

The threshold was chosen to be 0.27% (100–99.73) following the classic 68-95-99.7 empirical rule, or more exactly following that 99.73% of the observations fall within three standard deviations of the data, assumed to be normal distributed. Moreover, in the context of HMM, the anomalies are spotted by looking at observations with extremely low likelihoods (or those in which the model is not confident that they belong to any of the states), thus picking out the bottom 0.2% of those likelihoods yield potential outliers.

D. AUTOENCODER IMPLEMENTATION

In terms of implementation, a Bi-LSTM is used to build this network. The implementation of this model starts by splitting the data into groups (one for each cluster) and then an AE is applied for each group.

This AE is mainly composed of an Encoder and a Decoder. The Encoder consists of two Bi-LSTM layers, for which each LSTM cell has as output a vector of size 64 (and 32 respectively). The first layer outputs such a vector for each timestep (since the return sequence is set to True), while the second layer outputs a single final vector. Therefore, in order to match the dimension needed at the Decoder, a RepeatVector layer is further introduced in order to copy that final vector for each timestep. On the other hand, the Decoder mirrors the architecture outlined above for the Encoder. Afterwards, a TimeDistributed layer is used to restore the original number of features for each timestep.

Lastly, the parameters for training are set: the Mean Squared Error (MSE) loss function alongside the Adam optimizer. Then, the AE is trained for each working mode and further the model is used to predict the original data. The difference between the original and the predicted data enables us to detect anomalies, as significant differences can represent outliers found within the data.

Likewise, as with detecting anomalies using the HMM model, the AE yet again followed the classic empirical rule. More exactly, the observations where the squared difference between the original and the predicted output exceeded three standard deviations (or above 99.73% of all other observation likelihoods) were flagged as outliers. In contrast to HMM where small likelihood values resembled outliers, here the anomalies were considered those whose reconstruction error was large.

E. PSEUDO-LABELING TO VALIDATE ANOMALY DETECTION

In the context of this study, despite the unsupervised nature of the developed algorithms, an approach for pseudo-labeling the data using the Pauta criterion was employed to assess and compare their effectiveness [21], [22], [23]. This criterion is found by calculating the standard deviation (σ) and mean (μ) of the normalized dataset and flagging any data point (x) that deviates from the mean by a certain multiple (usually $k = 3$) of the standard deviation as an anomaly, as defined by the criterion. The identification of anomalies is realized by the following formula:

$$Anomaly = |x - \mu| > 3\sigma \tag{5}$$

In this expression, $|x - \mu|$ represents the absolute difference between a data point and the mean, and 3σ represents the threshold for flagging anomalies. Any data point exceeding this threshold is considered an anomaly and is subsequently labeled as such in the analysis. Although the sensors detected a wide range of data, in order to train the algorithms and measure the performance, the following means were used based on the three phases of the electric current: average reactive power, average power factor, average current, average voltage, average THDI, average THDU, Xacc, yaw and pitch.

V. RESULTS

We started the experiment by determining the operating modes of the factory machine using the HMM algorithm. Thus, first the scores used to automatically determine the number of hidden states were computed, namely BIC and ABIC. Fig. 4 illustrates that BIC does not stabilize and tends to suggest the model with the highest possible number of states. A lower value of BIC means that the model should perform better with that number of clusters, as it can be seen in Fig. 4.

In contrast, the ABIC coefficient penalizes the model harder, stabilizes faster and does not suggest choosing the

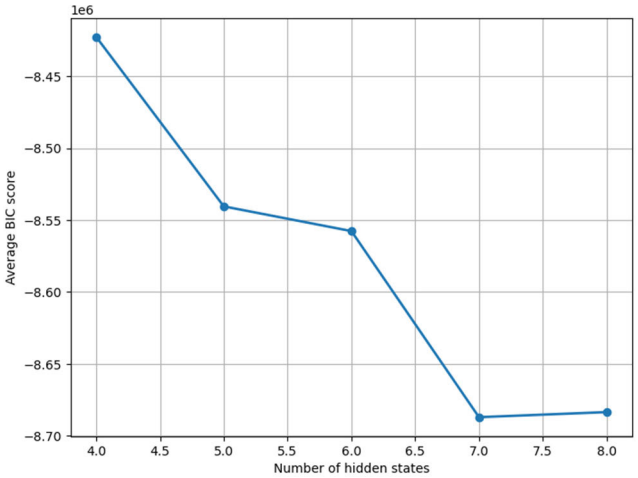


FIGURE 4. BIC score based on the number of hidden states within the HMM.

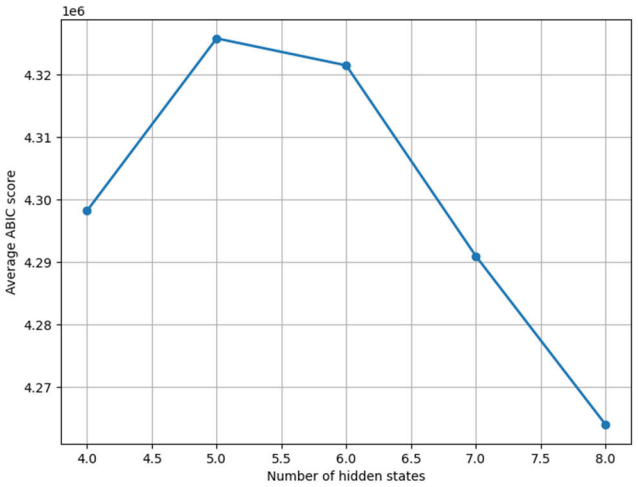


FIGURE 5. ABIC score based on the number of hidden states within the HMM.

highest possible number of hidden states. A higher value of ABIC means that the model should perform better with that number of clusters. This can be seen in Fig. 5, where ABIC suggests choosing 5 different clusters as a starting point for the current time series.

Since the nature of the problem is time series clustering, visual results tend to give better insights compared to the above metrics. Fig. 6 and 7 show some cases where the operating modes are visibly distinguished.

Moreover, the HMM model was also exploited in the context of detecting anomalies. Some samples of anomaly detection are shown in Fig. 8 and 9 (the anomalies are highlighted in the middle with a red rectangle).

The data was mapped to the [0,1] interval for a better visualization, also even though all features are taken into consideration by the HMM, only the feature which visually produced the anomaly was selected to be plotted.

Furthermore, to provide an efficient LSTM-based AE, it was essential to build an efficient architecture which learns

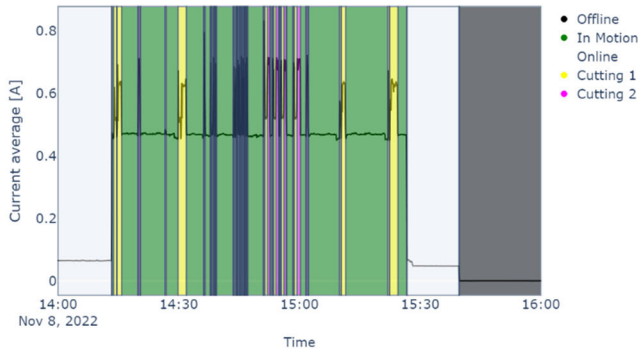


FIGURE 6. Example visualization for the detected modes on 8th November 2022 using HMM.

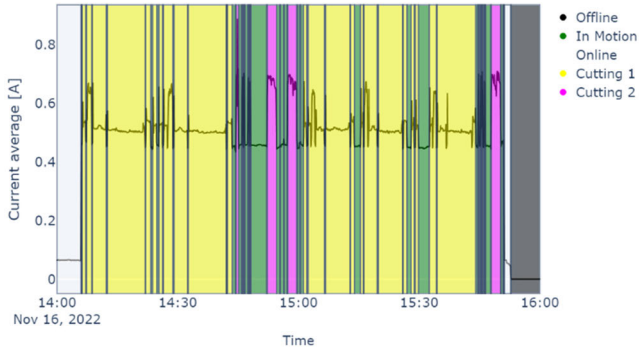


FIGURE 7. Example visualization for the detected modes on 16th November 2022 using HMM.

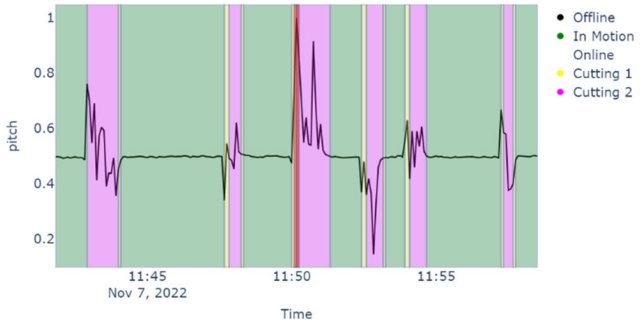


FIGURE 8. Anomaly detected by the HMM on pitch.

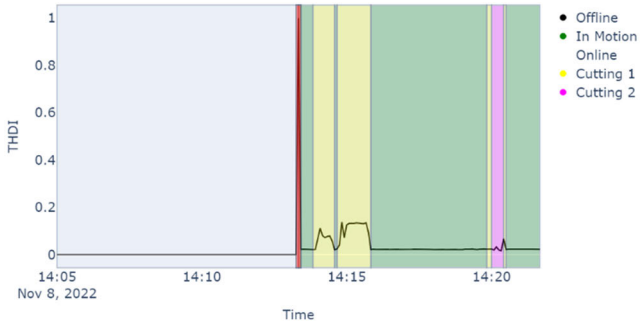


FIGURE 9. Anomaly detected by the HMM on THDI.

as best as possible to reconstruct the given time series. Thus, the MSE loss function was further employed to measure the model's performance.

In Table 1, different architectures, activation functions and output sizes were tested, with MSE being the loss function, whereas the rectified linear unit (ReLU) alongside the hyperbolic tangent (TANH) are the possible activation functions. As it was highlighted, the best performance was obtained by a Bi-LSTM-based AE that utilizes a TANH activation across four layers (with their output sizes at every timesteps being 64, 32, 32 and 64).

TABLE 1. MSE of AE.

Architecture	Activation function	Output sizes	MSE
Bi-LSTM	TANH	64×32×32×64	0.05984
Bi-LSTM	ReLU	64×32×32×64	0.10334
LSTM	TANH	64×32×32×64	0.19288
Bi-LSTM	TANH	32×32×32×32	0.08896
Bi-LSTM	TANH	64×32×16×16×32×64	0.07904

Fig. 10 and 11 illustrate cases where the aforementioned model detected some visual deviations within the data. In contrast to the HMM, where the anomalies are detected as the contributions of all the features within an observation, the AE detects anomalies for each individual feature.

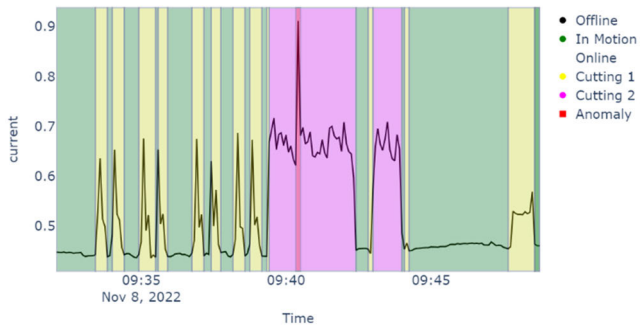


FIGURE 10. Anomaly detected by the AE on current.

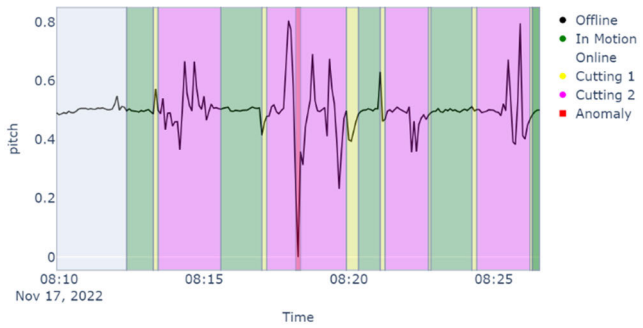


FIGURE 11. Anomaly detected by the AE on pitch.

Next, the predicted anomalies from our proposed HMM and AE methods and the state-of-the-art SES model [1] were compared with the instances which were pseudo-labeled using the Pauta criterion (as described in Section IV). In addition to the common metrics (recall, precision, accuracy, F1), some metrics specific to anomaly detection were also used, such as geometric mean (GM), Matthew's Correlation Coefficient (MCC) providing the quality of binary classification, as well as false alarm rate (FAR):

$$GM = \sqrt{\frac{TP}{TP+FN} \cdot \frac{FP}{FP+TN}} \tag{6}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}} \tag{7}$$

$$FAR = \frac{FP}{FP+TN} \tag{8}$$

Although five different operating modes were detected (Offline, Online, In Motion, Cutting 1 and Cutting 2), only In Motion, Cutting 1 and Cutting 2 were examined as the focus is on the operating modes where the machine is operating.

In Cutting 1, the HMM outperforms the AE and the SES in terms of recall, precision, F1 score, and GM. The HMM demonstrates a better ability to capture relevant instances (higher recall), maintain precision in positive predictions (similar precision) and achieve an overall balanced performance (higher F1 score and GM). The higher accuracy of the HMM further supports its effectiveness in this mode.

In Cutting 2, both the AE and HMM demonstrate strong performances in contrast with the SES algorithm, with the HMM having a slight edge in terms of recall and GM. The HMM achieves a better balance between sensitivity and specificity, resulting in a higher GM. While the AE excels in precision, the HMM still outperformed the AE as well as SES at the overall results.

When the machinery is In Motion, the AE outperforms the HMM and SES due to the higher precision and recall. This can be attributed to the fact that the HMM model has a high precision, but a low recall or, in other words, the model does not risk when predicting an anomaly, it only predicts when it is almost 100% sure that the observation will be an anomaly, while SES has trouble with finding the correct anomaly patterns.

We can also see from Tables 2, 3 and 4 that the MCC, used to determine the quality of binary classification, is achieving scores of 0.71 and 0.67 for HMM compared to 0.45 and 0.55 for the AE, In Motion being an exception. For the SES model, except for Cutting 1 (where it behaved similarly to the AE), the MCC scores are close to 0. Also, we can note that proposed approaches result in a low false alarm rate, making them suitable for industry usage. However, the HMM model is more in line with the pseudo-labeling performed through the Pauta criterion.

Table 4 showcases a decrease in performance for SES [1] compared to the AE and HMM. However, this outcome is expected given the two distinct architectures. Specifically, the AE's architecture was optimized for our problem (as highlighted in Table 1) to feature four LSTM layers (of size 64-32-32-64) whereas the SES utilized a single LSTM layer (with 20 units). In addition, the AE also aimed to capture

TABLE 2. Anomaly detection results for the AE.

Mode	Cutting 1	In Motion	Cutting 2
Recall	0.241185	0.264980	0.353448
Precision	0.924324	0.771318	0.911111
Accuracy	0.936654	0.948040	0.963544
GM	0.490676	0.513381	0.593935
F1	0.919093	0.935915	0.955816
MCC	0.454107	0.433346	0.554825
FAR	0.001749	0.005360	0.001950

TABLE 3. Anomaly detection results for the HMM.

Mode	Cutting 1	In Motion	Cutting 2
Recall	0.579690	0.006658	0.568966
Precision	0.915367	1.000000	0.835443
Accuracy	0.961441	0.936559	0.970928
GM	0.759564	0.081595	0.751904
F1	0.957421	0.906298	0.968299
MCC	0.711023	0.078964	0.675700
FAR	0.004747	0.000000	0.006338

TABLE 4. Anomaly detection results for the SES [1].

Mode	Cutting 1	In Motion	Cutting 2
Recall	0.250950	0.041958	0.10465116
Precision	0.048104	0.095238	0.03061224
Accuracy	0.827181	0.933697	0.83225208
GM	0.460533	0.202728	0.30042732
F1	0.774652	0.946102	0.79077449
MCC	0.045148	0.031901	-0.0187588
FAR	0.154849	0.020481	0.13754826

patterns backward within the data by utilizing bidirectional layers.

The increase in performance for the AE unfortunately also comes at a significant computational cost. At each timestep, the LSTM cell performs two matrix multiplications, for each of the three gates and the cell state update, leading to a complexity of $O(U^2 + U \cdot I)$, where U is the number of units and I is the input size. However, the AE with its bidirectionality increased the number of units and has four times more layers, resulting in approximately 16 times more computational demand than SES's simpler setup.

The HMM stands out not only for its impressive performance, but also for its slim complexity. Primary the most intensive part appears during the learning step, which is performed using the Baum-Welch algorithm and has a complexity of $O(H^2 \cdot N)$ for each iteration, where H is the number of hidden states and N is the time series length. The decoding part to assign a hidden state to every timestep and later employ the Forward algorithm in order to flag possible anomalies are also crucial, however, these two are performed only once over the time series, whereas the Baum-Welch algorithm iterates multiple times over the input sequence until the model converges. This highlights the effectiveness of the HMM approach in not only performing better in terms of performance, but also in terms of computational cost.

McNemar's test, a non-parametric pair-wise test, was conducted to assess the significance of differences between the performances of the HMM and AE algorithms. This test evaluates whether there is a statistically significant increase achieved by one algorithm over the other. The z-score is utilized to determine confidence levels. A z-value exceeding 1.96 ($p < 0.05$) indicates a significant difference between the two algorithms. In the context of McNemar's test, the z-score is calculated as follows:

$$z = \frac{|N_{12} - N_{21}| - 1}{\sqrt{N_{12} + N_{21}}} \tag{9}$$

where N_{12} represents the instances where the first algorithm classified correctly with the second algorithm classifying incorrectly and N_{21} represents the instances where the second algorithm classified correctly with the first algorithm classifying incorrectly. The results are presented in Table 5.

TABLE 5. Z-score based on the McNemar statistical significance test.

Algorithm	Cutting 1	In Motion	Cutting 2
HMM vs AE	26.19	0.32	14.68
AE vs SES	31.91	12.80	16.31
HMM vs SES	4.55	14.06	1.27

As Table 5 shows, comparing HMM and AE, reveals a significant difference in Cutting 1 and 2, while showing negligible variance in the In Motion scenario.

For AE versus SES, the analysis indicates substantial differences in all scenarios, as evidenced by high z-scores in Cutting 1 and 2, as well as in the In Motion situation.

Comparisons between HMM and SES show moderate to significant differences across scenarios, with varying z-scores with a quite low z-score in the Cutting 2 scenario.

VI. CONCLUSION AND FURTHER WORK

Throughout this study, a Machine Learning based software, consisting of a pipeline with two components, was applied for the CNC manufacturing data. The first component made the best out of the HMMs in order to perform Time Series Clustering, whereas the second component utilized both HMMs and LSTM-based AE in order to flag possible anomalies. More exactly, to cluster the energy consumption time series, several steps were taken.

For evaluation, a dataset was collected as a time series from a CNC machine and then the preprocessed data was further analyzed using the implemented software. First, the optimal

number of clusters was determined automatically using a novel coefficient that evaluated the model's performance, namely ABIC, which penalizes the model harder in order to stabilize and refrain from suggesting the highest possible number of hidden states.

Different metrics were applied in order to validate each model's performance and gain a better understanding of the ability to perform different tasks and outperform other algorithms. The AE naturally aimed to detect anomalies through its architecture, as it learned to reconstruct the time series in an encoder-decoder manner. Thus, the anomalies were identified as being those data points which, when reconstructed, presented major differences compared to the original ones. In terms of results, the HMM proved to be more consistent and robust compared to the AE and the state-of-the-art SES method [1], especially in achieving higher F1 and MCC scores. This highlights its potential for practical applications where accurate anomaly detection is crucial for ensuring the reliability and efficiency of machine operations. Consequently, by assuring a fast automatic working mode clustering and anomaly detection, the proposed methodology addressed the need for developing a sustainable manufacturing, as it facilitates the analysis of the data obtained from the afferent machines.

In addition, the proposed algorithms can provide a starting point for the application of other techniques, which aim to improve the environmental impact in this industry, or in other related fields as well. One possible extension for the existing software could be to develop a prediction mechanism in order to preemptively alarm future anomalies. Algorithms such as the autoregressive integrated moving average model, or even those that were already applied in this study (HMMs and LSTMs) can be taken into consideration. We plan to integrate the best anomaly detection method into a real woodworking factory environment.

REFERENCES

[1] Z. Ji, Y. Wang, K. Yan, X. Xie, Y. Xiang, and J. Huang, "A space-embedding strategy for anomaly detection in multivariate time series," *Exp. Syst. Appl.*, vol. 206, Nov. 2022, Art. no. 117892.

[2] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, 2014.

[3] M. Achouch, M. Dimitrova, K. Ziane, S. S. Karganroudi, R. Dhouib, H. Ibrahim, and M. Adda, "On predictive maintenance in Industry 4.0: Overview, models, and challenges," *Appl. Sci.*, vol. 12, no. 16, p. 8081, Aug. 2022.

[4] A. Bécue, I. Praça, and J. Gama, "Artificial intelligence, cyber-threats and Industry 4.0: Challenges and opportunities," *Artif. Intell. Rev.*, vol. 54, no. 5, pp. 3849–3886, Jun. 2021.

[5] T. Zheng, M. Ardolino, A. Bacchetti, and M. Perona, "The applications of Industry 4.0 technologies in manufacturing context: A systematic literature review," *Int. J. Prod. Res.*, vol. 59, no. 6, pp. 1922–1954, Mar. 2021.

[6] M. Gebhardt, M. Kopyto, H. Birkel, and E. Hartmann, "Industry 4.0 technologies as enablers of collaboration in circular supply chains: A systematic literature review," *Int. J. Prod. Res.*, vol. 60, no. 23, pp. 6967–6995, Dec. 2022.

[7] W. Jiang, "A machine vision anomaly detection system to Industry 4.0 based on variational fuzzy autoencoder," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–10, Mar. 2022.

[8] G. P. Tancredi, G. Vignali, and E. Bottani, "Integration of digital twin, machine-learning and Industry 4.0 tools for anomaly detection: An application to a food plant," *Sensors*, vol. 22, no. 11, p. 4143, May 2022.

[9] T. Sahar, M. Rauf, A. Murtaza, L. A. Khan, H. Ayub, S. M. Jameel, and I. U. Ahad, "Anomaly detection in laser powder bed fusion using machine learning: A review," *Results Eng.*, vol. 17, Mar. 2023, Art. no. 100803.

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

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

[12] J. Reed and C.-H. Lee, "A study on music genre classification based on universal acoustic models," in *Proc. 7th Int. Conf. Music Inf. Retr.*, 2006, pp. 89–94.

[13] P. Smyth, "Clustering sequences with hidden Markov models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1996, pp. 1–7.

[14] J. Pohle, R. Langrock, F. M. van Beest, and N. M. Schmidt, "Selecting the number of states in hidden Markov models—Pitfalls, practical challenges and pragmatic solutions," *J. Agricult., Biol., Environ. Statist.*, vol. 22, no. 3, pp. 270–293, 2017.

[15] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–38, 2021.

[16] K. Aditya, H. Kolla, W. P. Kegelmeyer, T. M. Shead, J. Ling, and W. L. Davis, "Anomaly detection in scientific data using joint statistical moments," *J. Comput. Phys.*, vol. 387, pp. 522–538, Jun. 2019.

[17] M. Hu, X. Feng, Z. Ji, K. Yan, and S. Zhou, "A novel computational approach for discord search with local recurrence rates in multivariate time series," *Inf. Sci.*, vol. 477, pp. 220–233, Mar. 2019.

[18] J. M. Garcia, T. Navarrete, and C. Orozco, "Workload hidden Markov model for anomaly detection," *Proc. Int. Conf. Secur. Cryptography*, Aug. 2006, pp. 56–60.

[19] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Paris, France, Oct. 2015, pp. 1–7.

[20] S. Lee, Y. Yun, S. Park, S. Oh, C. Lee, and J. Jeong, "Two phases anomaly detection based on clustering and visualization for plastic injection molding data," *Proc. Comput. Sci.*, vol. 201, pp. 519–526, Jan. 2022.

[21] A. Aligholian, M. Farajollahi, and H. Mohsenian-Rad, "Unsupervised learning for online abnormality detection in smart meter data," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Atlanta, GA, USA, Aug. 2019.

[22] S. Wang, Z. Zhang, P. Wang, and Y. Tian, "Failure warning of gearbox for wind turbine based on 3σ-median criterion and NSET," *Energy Rep.*, vol. 7, pp. 1182–1197, Nov. 2021.

[23] C. Li, L. Guo, H. Gao, and Y. Li, "Similarity-measured isolation forest: Anomaly detection method for machine monitoring data," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.

[24] A. Gellert and L. Vintan, "Person movement prediction using hidden Markov models," *Stud. Inform. Control*, vol. 15, no. 1, p. 17, 2006.

[25] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Jul. 2023.

[26] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[27] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," 2021, *arXiv:2106.11342*.

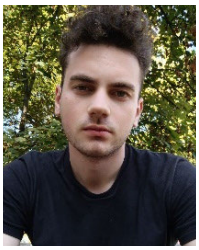
[28] S.-A. Precup, A. Gellert, A. Dorobantiu, and C.-B. Zamfirescu, "Assembly process modeling through long short-term memory," in *Proc. 13th Asian Conf. Intell. Inf. Database Syst.*, Phuket, Thailand, Apr. 2021, pp. 28–39.

[29] A. Matei, S.-A. Precup, D. Circa, A. Gellert, and C.-B. Zamfirescu, "Estimating travel time for autonomous mobile robots through long short-term memory," *Mathematics*, vol. 11, no. 7, p. 1723, Apr. 2023.

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

[31] A. Karpathy. (2015). *The Unreasonable Effectiveness of Recurrent Neural Networks*. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
[32] A. Kramer, "Autoassociative neural networks," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 313–328, 1992.
[33] H. D. Trinh, E. Zeydan, L. Giupponi, and P. Dini, "Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach," *IEEE Access*, vol. 7, pp. 152187–152201, 2019.

RADU SOROSTINEAN received the B.Sc. degree in computer science from the University of Southampton and the M.Sc. degree in advanced computing systems from Lucian Blaga University of Sibiu. He is currently pursuing the Ph.D. degree in sustainable production systems. He has noteworthy contributions to AI-based assistance systems within the context of Industry 4.0, with a focus on machine learning and AI. He has significant contributions to projects for famous automotive companies and energy initiatives in South America.



ZAHARIA BURGHELEA received the B.Sc. degree in computer science and the M.Sc. degree in advanced computing systems from Lucian Blaga University of Sibiu. He is currently a Software Engineer for an IoT company, focusing on innovative technology solutions. His research interests include machine learning and quantum computing, where he actively studies and develops various projects. In addition to this, he also contributed with problems and solutions in a variety of Math journals and magazines.

ARPAD GELLERT received the M.Sc. and Ph.D. degrees in computer science from Lucian Blaga University of Sibiu (LBUS), in 2003 and 2008, respectively, and the Habilitation degree in computers and information technology, in 2023. He is currently an Associate Professor with the Computer Science and Electrical Engineering Department, LBUS. He published seven books and more than 40 scientific papers in prestigious journals and international conferences indexed in *Web of Science*. His works were cited in more than 400 different publications. He was a member of several research grants. As the Project Manager, he developed a research grant supported by the Romanian National Council of Academic Research (CNCSIS). Recently, he was the Manager of a Horizon 2020 Research Project, in a partnership with the CENTROPLAST factory from Serbia, financed by DIH-World. He was also the Manager of five internal LBUS grants. He is the Manager of an ongoing Hasso Plattner Excellence Research Grant. His research interests include computer architecture, smart buildings, smart factories, web mining, and image processing. He received the "Ad Augusta Per Angusta" prize awarded by Lucian Blaga University of Sibiu for excellence in scientific research, in 2010.