energies

MDPI

Article

LSTM-Autoencoder Deep Learning Model for Anomaly Detection in Electric Motor

Fadhila Lachekhab [1,*], Messouada Benzaoui [2], Sid Ahmed Tadjer [3], Abdelkrim Bensmaine [1] and Hichem Hamma [1]

[1] Applied Automation Laboratory, Faculty of Hydrocarbon & Chemistry, University M'Hamed Bougara, Boumerdes 35000, Algeria; akrambensmaine27@gmail.com (A.B.); hichemhamma28@gmail.com (H.H.)
[2] Applied Automation Laboratory, Institute of Electrical and Electronic Engineering, University M'Hamed Bougara, Boumerdes 35000, Algeria; m.benzaoui@univ-boumerdes.dz
[3] Electrification of Industrial Enterprises Laboratory, Faculty of Hydrocarbon & Chemistry, University M'Hamed Bougara, Boumerdes 35000, Algeria; s.tadjer@univ-boumerdes.dz
* Correspondence: f.lachekhab@univ-boumerdes.dz; Tel.: +213-6-66-87-65-52

Abstract: Anomaly detection is the process of detecting unusual or unforeseen patterns or events in data. Many factors, such as malfunctioning hardware, malevolent activities, or modifications to the data's underlying distribution, might cause anomalies. One of the key factors in anomaly detection is balancing the trade-off between sensitivity and specificity. Balancing these trade-offs requires careful tuning of the anomaly detection algorithm and consideration of the specific domain and application. Deep learning techniques' applications, such as LSTMs (long short-term memory algorithms), which are autoencoders for detecting an anomaly, have garnered increasing attention in recent years. The main goal of this work was to develop an anomaly detection solution for an electrical machine using an LSTM-autoencoder deep learning model. The work focused on detecting anomalies in an electrical motor's variation vibrations in three axes: axial (X), radial (Y), and tangential (Z), which are indicative of potential faults or failures. The presented model is a combination of the two architectures; LSTM layers were added to the autoencoder in order to leverage the LSTM capacity for handling large amounts of temporal data. To prove the LSTM efficiency, we will create a regular autoencoder model using the Python programming language and the TensorFlow machine learning framework, and compare its performance with our main LSTM-based autoencoder model. The two models will be trained on the same database, and evaluated on three primary points: training time, loss function, and MSE anomalies. Based on the obtained results, it is clear that the LSTM-autoencoder shows significantly smaller loss values and MSE anomalies compared to the regular autoencoder. On the other hand, the regular autoencoder performs better than the LSTM, comparing the training time. It appears then, that the LSTM-autoencoder presents a superior performance although it was slower than the standard autoencoder due to the complexity of the added LSTM layers.

Keywords: long short-term memory algorithm; deep learning model; autoencoder model; anomaly detection; electrical machine

check for updates

Citation: Lachekhab, F.; Benzaoui, M.; Tadjer, S.A.; Bensmaine, A.; Hamma, H. LSTM-Autoencoder Deep Learning Model for Anomaly Detection in Electric Motor. *Energies* **2024**, *17*, 2340. https://doi.org/10.3390/en17102340

Academic Editor: Youguang Guo

Received: 22 February 2024
Revised: 4 May 2024
Accepted: 7 May 2024
Published: 13 May 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Since the 18th century's first industrial revolution, industrial maintenance techniques have evolved to address the challenges of equipment reliability and performance in industrial settings. Initially, the predominant approach was reactive maintenance [1], also known as "breakdown maintenance". This involved waiting for equipment failures to occur and then taking corrective actions to fix the issues. While this approach was simple and cost-effective in the short term, it often resulted in significant production losses, safety hazards, and higher repair costs.

As industries became more complex and downtime costs increased, predictive maintenance emerged as a more proactive strategy, involving routine inspections and maintenance

tasks based on predetermined schedules. This approach aimed to prevent unexpected failures by addressing known wear and tear issues. While preventive maintenance reduced unplanned downtime to some extent, it was not always efficient and often led to unnecessary maintenance activities and associated costs.

In recent years, with advancements in technology and the rising implementation of artificial intelligence techniques in the industrial sector, predictive maintenance (PdM) has gained prominence [2]. This approach utilizes real-time data from sensors, monitoring systems, and predictive algorithms to assess equipment conditions, identify potential failures, and schedule maintenance activities accordingly. By adopting this approach, organizations can optimize maintenance schedules, reduce costs, maximize equipment uptime, and enhance overall operational efficiency.

Furthermore, artificial intelligence, artificial neural networks (ANNs), and deep learning (DL) methods are being integrated into industrial maintenance practices [3,4]. These technologies enable more advanced data processing, anomaly detection, and predictive modeling, leading to more accurate predictions and optimized maintenance strategies. One of the most common DL techniques is the long short-term memory (LSTM) architecture, which is one of the recurrent neural network algorithms (RNN) that can model and predict sequential data [5]. RNN can be used as a regression model for anomaly detection. The main issue with the standard RNN is its inability to learn long-term patterns in sequential data due to the gradient vanishing/exploding problem when applying a backpropagation-through-time (BPTT) algorithm during the training phase. For this reason, a standard RNN is rarely used in real world applications, which are usually based, instead, on two improved RNN variants: the long short-term memory (LSTM) and the gated recurrent unit (GRU).

Barso presents a survey of recent advances in anomaly detection methods such as convolutional neural networks (CNN), generative models, variational autoencoder (VAE), and temporal logic-based learning [6].

Predictive maintenance is a very advanced maintenance technique, it requires a significant amount of data to properly function and predict future failures before they occur. In order to properly handle the large amount of data required and extract patterns to perform efficient training and predict future failures, the implementation of DL and specifically LSTM-autoencoder models might be essential.

LSTM-autoencoders can capture long-term dependencies and model contextual information, making them particularly useful for tasks involving sequential data with temporal dynamics [7]. Thus, LSTM is perfectly suited to treat, train, and learn from our database, which contains a large number of sequential vibration data.

Our model presents a combination of the two architectures; LSTM layers were added to the autoencoder in order to leverage the LSTM capacity to manage large amounts of temporal data inputs.

To prove the model efficiency, we first introduced a regular autoencoder and trained both models on the same data using Python. After visualizing the results and competence of the two models, we compared and reviewed their performance on three points: training time, loss function, and MSE anomalies.

2. Related Work

Predictive maintenance (PdM) methods, based on artificial intelligence (AI) techniques such as deep learning (DL) and machine learning (ML), have recently been widely used in industries to manage the health status of industrial processes. As a result of the development and increasing popularity of these DL algorithms, now it is possible to gather enormous amounts of operational and process conditions data generated from various pieces of equipment and use the data to make an automated fault detection, diagnosis, and most importantly, a prognosis to increase the utilization rate of the components and reduce and predict downtime [8].

The paper published in [9] pinpointed the current landscape of AI in manufacturing. A systematic review of different journals and science source materials was made to under-

stand better the requirements and necessary steps for a successful transition into Industry 4.0 supported by AI and the challenges that may occur during this process.

A state-of-the-art analysis of the ongoing and upcoming AI research is given by Zhang in [10]. Noting that AI is a multidisciplinary field with various applications in numerous domains, it concluded that the next advancement in this field can not only provide computers with better logical thinking powers but can also give them emotional capabilities. It is possible that soon machine intelligence may surpass human intelligence. One of the key drivers of artificial intelligence is machine learning (ML), which focuses on developing algorithms that can learn from data and improve their performance over time.

Without the need for more explicit programming, machine learning algorithms arrange the data, learn from them, acquire insights, and generate predictions based on the information they analyze [11]. Machine learning is concerned with using data to train a model and then using the model to predict any incoming data [12].

The significant advantage of machine learning is its capacity to handle complex and large-scale datasets. By processing vast amounts of data, machine learning algorithms can uncover intricate patterns and relationships that may not be obvious to humans. This enables applications in various domains, such as speech recognition and image, natural language processing, recommendation systems, fraud detection, autonomous vehicles. . . etc.

Computer science, including AI and distributed computing areas, is increasingly prominent in a field where engineering is predominant, highlighting the necessity of a multidisciplinary methods to properly meet Industry 4.0. However, several restrictions and obstacles are categorizing this field [8].

The historical overview of maintenance was mentioned. In the article, the potential for a "new" kind of maintenance associated with Industry 4.0, namely PdM, was proposed. They concluded that PdM, being the most advanced form of all maintenance, is what companies strive to develop and what can give them an advantage over others.

ML algorithms have been widely used in computer science and other fields, including PdM of production systems, tools, and machines, it is one of the potential applications for data-driven approaches. ML algorithms can solve many problems using the important number of available data created by industries. In a study about the recent advancements of ML techniques applied to PdM, the most commonly used ML algorithms for PdM were those mentioned in [13]: logistics regression (LR), support vector machine (SVM), reinforcement learning (RL), and decision tree (DT). The continuous growth of PdM was highlighted [14]. Ref. [15] surveyed papers related to the automotive industry from an ML perspective, mentioning the adequacy of ML for PdM, and concluding that the implementation of DL techniques will increase but requires the availability of large amounts of labeled data. Ref. [16] is an article that aims at facilitating the task of choosing the right DL model for PdM by reviewing cutting-edge DL architectures and how they integrate with PdM to satisfy the needs of industrial companies (anomaly detection, root cause analysis, and remaining useful life estimation). They are categorized in industrial applications, with an explanation of how to close any gaps. Open difficulties and potential directions for further research are then outlined. Ref. [17] is an article summarizing the fundamentals of ML and DL to generate a broader understanding of the systematic framework of current intelligent systems. The authors abstractly defined keywords and concepts, described how to develop automated analytical models using ML and DL, and discussed the difficulties in applying such intelligent systems in the context of electronic marketplaces and networked commerce.

Machine learning techniques can be subdivided into supervised, unsupervised, semi-supervised, and reinforcement learning. Ref. [18] is an article that reviews the state of supervised learning research, concentrating on three common forms of weak supervision: incomplete supervision, inexact supervision, and inaccurate supervision. It was determined that when there is a multitude of training instances with ground-truth labels, supervised learning techniques have had remarkable success. However, in practical applications, gathering supervision information incurs costs, making the ability to perform weakly

supervised learning often beneficial [19]. The authors of [20] describe semi-supervised learning as a field. The survey provides an up-to-date analysis of this crucial area of ML, covering techniques from the early 2000s as well as more recent developments. Additionally, they have introduced a new taxonomy for semi-supervised categorization techniques that makes distinctions between the approach's main goal and unlabeled data.

The training and learning of ANN-based unsupervised learning is outlined in [21], where they explain the procedures for choosing and fixing several hidden nodes in an ANN-based unsupervised learning environment. Additionally, a summary of the status, advantages, and difficulties of unsupervised learning are described. A manuscript introducing deep RL models, algorithms, and techniques is published in [22].

Focusing in particular on the generalization features and the practical uses of deep RL, one of which is the CNN algorithm. The authors of [23] wrote a paper that provides a thorough analysis of the fundamental design ideas and technical uses of 1D CNNs, with a particular emphasis on current advancements in this area. Finally, their distinctive qualities are highlighted, capping off their cutting-edge performance. Ref. [23] proposed a data-driven approach that combines RNNs with graspable explanations for predicting the probability of mortality. This method was able to identify and clarify the historical contributions of the linked elements to the prediction, in addition to providing the anticipated mortality risk. It was determined that if patients' clinical observations in the ICU are continually monitored in real time, they may benefit from early intervention.

A survey on RNNs and several new advances for newcomers was exposed in [24] and by professionals in the field. The fundamentals and recent advances are explained and the research challenges are introduced, mentioning other RNN architectures, especially LSTM.

3. Long Short-Term Memory Algorithm

A typical feature of RNN architecture is cyclic connectivity, which gives the RNN the ability to update the current state based on past states and current input data. These networks, consisting of standard recurrent cells, have had incredible success with numerous challenges. Unfortunately, when the gap between the relevant input data is large, the above RNNs are unable to connect the relevant information [25].

To handle the "long-term dependencies", Hochreiter and Schmidhuber [26] proposed the long short-term memory (LSTM) model.

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) that is particularly useful for working with sequential data, such as time series anomaly detection, which makes it convenient to implement in the PdM approach.

In particular, LSTMs excel at handling the complex and dynamic nature of equipment performance data, which often contains multiple variables and dependencies. By capturing the long-term dependencies in the data, LSTMs can provide more accurate predictions of future equipment failures, enabling organizations to take preventive measures before failures occur [6].

The LSTM has become the focus of DL. To investigate its learning capacity, the authors in [25] examined the LSTM cell and its variations. They also divided LSTM networks into two primary types, LSTM-dominated networks and integrated LSTM networks, and discussed their different applications. Finally, LSTM network research directions were outlined. The training process of RNNs was reviewed in [27], and the authors explained how the LSTM neural networks can handle the main weakness of RNNs by learning long-term dependencies. In [28], an RNN-LSTM sentiment analysis model was put forth. To provide structured knowledge that can be applied to certain tasks, the goal was to build systems capable of extracting subjective information from natural language documents, such as feelings and opinions. With a 96% success rate, the LSTM model's performance was quite remarkable. The LSTM networks can learn higher-level temporal patterns without prior knowledge of the pattern duration, and it may be a practical method to model typical time series behavior, which can be used to detect anomalies [29].

A specific type of neural network is the autoencoder algorithm. The autoencoders' architecture, goals, and different applications were mentioned in [30].

To identify between spam reviews and legitimate reviews, an unsupervised learning model integrating LSTM networks and an autoencoder (LSTM-autoencoder) was suggested in [31]. The model in question was trained on how to identify real review trends from textual details only. The experimental findings demonstrate that the model can distinguish between legitimate and spam reviews with reasonable accuracy.

In engineering applications, an LSTM-autoencoder model was utilized for training and testing to improve the accuracy of the anomaly detection procedure [32]. This strategy enabled the identification of patterns and trends in the vibration data that might not have been obvious when using more conventional techniques. The accuracy percentage for finding anomalies in the vertical carousel system using the correlation coefficient model and LSTM-autoencoder was 97%.

The study presented in [33] proposed an LSTM network-based approach for multivariate time series data forecasting, in addition to an LSTM autoencoder network-based approach coupled with a one-class SVM algorithm for anomaly detection in sales. The acquired results demonstrate that, in comparison to the LSTM-based method proposed in prior work, the LSTM autoencoder-based method leads to improved performance for anomaly identification.

The LSTM architecture presented in Figure 1 consists of several units, each containing three main components: the input gate, the forget gate, and the output gate. These gates work together to control the flow of information into and out of the memory cell [6].

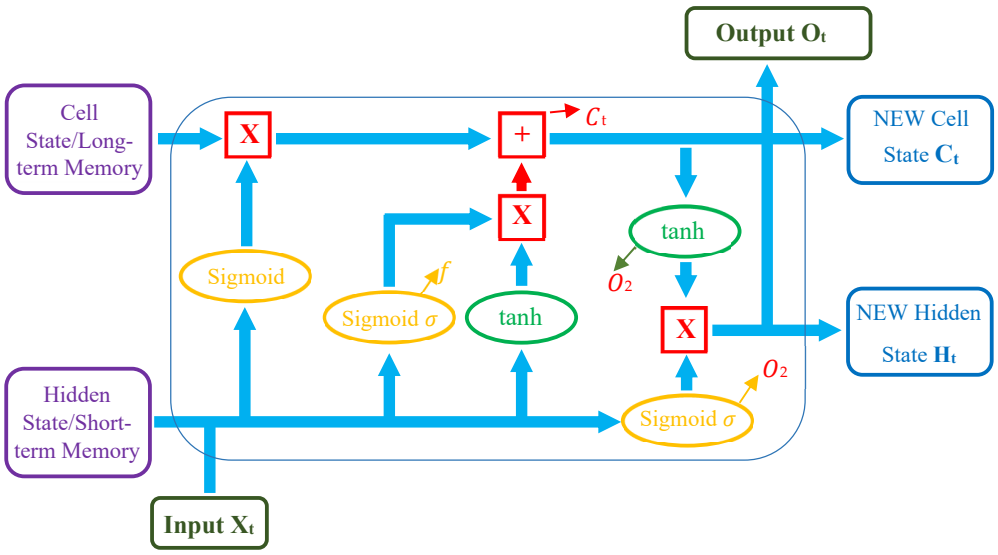


Figure 1. LSTM gates' diagram [5].

3.1. The Input Gate

The input gate determines which information is relevant to the current time step and should be stored in the memory cell. It takes as input the current input and the previous hidden state and applies an activation function (typically a sigmoid function) to each component. The sigmoid function is commonly used in neural networks as an activation function for binary classification problems [34].

The calculations of the first layer can be represented by the following equation:

$$i_1 = \sigma(W_{i1} \cdot (H_{t-1}, x_t) + bias_{i1}) \quad (1)$$

where W_{i1} is the weight matrix of the first layer, i_1 and H_{t-1} is the previous hidden state, x_t is the current input, and $bias_{i1}$ is a vector added to improve the accuracy of the model.

The second layer represents the calculation of the candidate values, regulating the network by passing the previous hidden state and current input into the hyperbolic tangent function, as follows:

$$i_2 = \tanh(\mathrm{W_{i2}} \cdot (\mathrm{H_{t-1}}, \mathrm{x_t}) + \mathrm{bias_{i2}}) \quad (2)$$

The outputs of these two layers are then multiplied and the information that needs to be stored in the memory cell results:

$$i_{input} = i_1 \cdot i_2 \quad (3)$$

3.2. The Forget Gate

It determines which information in the memory cell should be forgotten or discarded, based on the current input and the previous hidden state. Its main role is to prevent the network from remembering irrelevant or outdated information, which could lead to overfitting or poor performance.

To achieve this, the LSTM's forget gate calculates a forget vector, which is a set of values between 0 and 1 that determine how much of each element in the previous long-term memory should be preserved or forgotten. The forget vector is created by passing the concatenation of the current input and the previous short-term memory through a sigmoid activation function. This sigmoid function maps the input to a range between 0 and 1, similarly to the input gate, with values closer to 0 indicating that the corresponding element in the previous long-term memory should be forgotten, and values closer to 1 indicating that the element should be preserved.

The forget vector has values ranging from 0 to 1 and can be mathematically represented by the following equation:

$$f = \sigma\Big(\mathrm{W_{forget}}(\mathrm{H_{t-1}}, \mathrm{x_t}) + \mathrm{bias_{forget}}\Big) \quad (4)$$

Once the forget vector is calculated, it is multiplied element-wise by the previous long-term memory to obtain the new long-term memory, as follows:

$$C_t = f \odot C_{t-1} \quad (5)$$

where C_t is the new long-term memory, f is the forget vector, $\odot$ represents the element-wise multiplication, and C_{t-1} is the previous long-term memory.

The new long-term memory is then updated with the information from the current input using the input gate, which determines which parts of the current input should be added to the long-term memory.

$$C_t = f \odot C_{t-1} + i_{input} \quad (6)$$

This process effectively erases information from the previous long-term memory that is no longer relevant to the current input. By doing so, the network can learn to focus on the most important features of the input data and make better predictions or decisions.

3.3. The Output Gate

The output gate in an LSTM cell is a key component that determines which parts of the long-term memory and current input are passed on to the next cell or used as the final output of the network. It is responsible for regulating the flow of information and selectively passing on relevant information to subsequent time steps or as output [35].

The output gate takes as input the current input, the previous hidden state, and the current long-term memory, which have all been processed by their respective gates (input and forget gates), as previously explained.

First, the current input and the previous hidden state are passed into the sigmoid activation function with the appropriate weights, which will determine the proportion of the current long-term memory that should be included in the new short-term memory.

$$O_1 = \sigma(W_{O1}(H_{t-1}, x_t) + bias_{O1}) \quad (7)$$

Then, the tanh activation function is applied to the new long-term memory, which was calculated by the forget gate and updated by the input gate. This normalizes the values of the new long-term memory.

$$O_2 = \tanh(W_{O2} \cdot C_t + bias_{O2}) \quad (8)$$

The normalized new long-term memory is then multiplied element-wise with the output of the sigmoid gate to produce the new short-term memory:

$$H_t \cdot O_t = O_1 \odot O_2 \quad (9)$$

The hidden state/short-term memory and cell state/long-term memory produced by these gates is then passed to the next time step for the process to be repeated or used as the final output of the network.

4. System and Database Description

Before we introduce our model, we need to clarify the source and characteristics of the used data and system. Our database is composed of time series data collected from sensors installed on SpectraQuest's Machinery Fault Simulator (MFS) Alignment-Balance-Vibration (ABVT) system.

SpectraQuest (Richmond, VA, USA) is a company that specializes in providing solutions for machinery fault diagnosis, condition monitoring, and vibration analysis. They offer a range of products and services aimed at helping industries ensure the reliability, performance, and safety of their machinery and equipment.

The SpectraQuest's MFS ABVT is a specialized piece of equipment designed to simulate various fault conditions and performance scenarios in machinery (Table 1). It is commonly used for research, testing, and training purposes in the field of fault diagnosis and condition monitoring.

To collect the data, four sensors were used:

- Three Industrial IMI Sensors, Model 601A01 accelerometers on the radial, axial, and tangential directions: sensibility (±20%) 100 mV per g (10.2 mV per m/s^2). Frequency range (±3 dB) 16-600000 CPM (0.27–10.000 Hz). Measurement range ±50 g (±490 m/s^2).
- One IMI Sensors triaxial accelerometer, Model 604B31, returning data over the radial, axial, and tangential directions: sensibility (±20%) 100 mV per g (10.2 mV per m/s^2). Frequency range (±3 dB) 30-300000 CPM (0.5–5.000 Hz). Measurement range ±50 g (±490 m/s^2).

The used characteristics of the MFS ABVT are given in the following table:

Table 1. Specifications of the MFS ABVT [36].

Specification	Value
Motor	1/4 CV DC
System weight	22 Kg
Frequency range	700–3600 rpm
Rotor	15.24 cm
Diameter of axis	16 mm
length of axis	520 mm
Bearings distance	30 mm
Balls number	8
diameter of balls	0.7145 cm
Cage diameter	2.8519 cm
FTF	0.3750 CPM/rpm

Our database contains two simulated states:

- Normal functioning state: this state represents the normal operating condition of the machinery, where all components are functioning properly and there are no faults or abnormalities.
- Imbalance state: this state simulates an imbalance in the rotating components of the machinery by adding weights ranging from 6 g to 35 g. Imbalance can occur due to uneven distribution of mass, leading to vibrations and performance issues.

In the dataset, there are 49 normal sequences without any faults. Each normal sequence corresponds to a fixed rotation speed ranging from 737 rpm to 3686 rpm, with an increment of approximately 60 rpm between each sequence.

For the imbalance sequences, the same 49 rotation frequencies used in the normal operation case are employed for loads below 30 g. However, for loads equal to or above 30 g, the resulting vibrations make it impractical for the system to achieve rotation frequencies above 3300 rpm. This limitation reduces the number of distinct rotation frequencies and measurements available. To conclude, we used in our program a simulated database obtained from SpectraQuest's Machinery Fault Simulator.

In anomaly detection, the goal is to identify patterns in data that deviate significantly from what is considered normal. Anomaly detection using LSTM networks is particularly effective for time series data, where patterns can change over time and may be difficult to detect using traditional methods [37].

To use LSTM anomaly detection, the first step is to train an LSTM network on normal data to learn the patterns and relationships in the time series. This training process involves feeding the LSTM network with historical data and optimizing the network's parameters to minimize the difference between the predicted and actual values. Once the network has been trained on normal data, it can be used to detect anomalies in new data.

When the LSTM encounters a time series data point that deviates significantly from the learned patterns, it can flag that data as anomalous and alert the user about potential issues. For example, in the context of predictive maintenance, an LSTM network trained on sensor data from industrial equipment can identify patterns that indicate potential equipment failures, allowing maintenance teams to take proactive measures to prevent downtime.

Autoencoders are a type of neural network that can learn to encode and decode different types of data. Commonly used in unsupervised learning tasks, the goal of an autoencoder is to learn a compressed representation of the input data in a lower-dimensional space, and then use this representation to reconstruct the original data as accurately as possible [30].

The autoencoder consists of two main parts: an encoder and a decoder. The encoder takes the input data and maps it to a lower-dimensional latent space, while the decoder takes the encoded data and reconstructs the original input data.

By training the network to minimize the difference between the input data and the reconstructed data, the autoencoder can learn to capture the most important features of the input data and ignore any irrelevant or noisy information.

Autoencoders have a wide range of applications, including data compression, image and speech recognition, and anomaly detection [30].

In anomaly detection, autoencoders can be used to identify patterns in temporal data by learning to encode the normal behavior of a system. The idea is to train the autoencoder on a dataset of normal, or non-anomalous, instances, and then use it to reconstruct new instances. When an anomalous instance is encountered, it will likely have a higher reconstruction error than normal instances, since it does not fit the learned pattern. Thus, the reconstruction error can be used as a metric for anomaly detection, and instances with high reconstruction errors can be flagged for further investigation.

Autoencoders have several advantages over traditional anomaly detection methods; they can learn complex patterns in data and do not require explicit feature engineering. They are also able to adapt to new and changing patterns in the data, making them suitable for dynamic systems.

4.1. Data Split

To effectively train our model, the split of the data into a training set and a test set is necessary:

- The training set is the portion of the dataset on which the model learns the underlying patterns and relationships between the input features and the target variable. The training set is typically larger than the test set to provide enough data for the model to learn from.
- The test set is a subset of the data that are used to evaluate the performance of the trained model. It serves as an unseen dataset that the model has not been exposed to during the training phase. The test set is used to assess how well the model generalizes to new, unseen data. By making predictions on the test set, the model's performance metrics, such as accuracy, precision, recall, or mean squared error, can be evaluated. The test set helps to determine the effectiveness and reliability of the trained model.

The shape of the training set and split set vary according to the amount of data used. In general, the larger the quantity of data the greater percentage we can use for the training set. In the case of this study, 12 million data values are used, which allowed us to perform a 95% to 5% train–test split.

4.2. Data Preprocessing

Preprocessing is a necessary step applied to transform the raw data into a format that maximizes the performance and reliability of our model.

Down-sampling is a preprocessing technique often used when the dataset is too large to be handled effectively. It aims to reduce the amount of data treated in order to create a more balanced dataset, which can improve the performance and fairness of the DL model. In this study, the "downSampler" function was used. It is an implementation of a down-sampling technique that reduces the size of the dataset by calculating the mean of consecutive subsets of the data. the train and test data are down-sampled with a sampling rate of 1000, which reduces the size of both datasets by aggregating consecutive subsets of 1000 samples into single rows.

LSTM models are primarily designed to work with a three-dimensional data format, thus, to benefit from the full potential of the proposed model (time series anomaly detection), the reshaping of the 2D data into 3D data is the last step of the preprocessing of data.

5. LSTM-Autoencoder

In the LSTM autoencoder model architecture, the input data X are passed through several LSTM layers. The first LSTM layer (L1) processes the input data, returning sequences to preserve the temporal information. The second LSTM layer (L2) further processes the output from the first layer, but does not return sequences. Instead, it compresses the information into a fixed-length vector. Line L3 creates a layer that repeats the compressed representation of the input sequence, allowing subsequent LSTM layers to process it and generate a reconstructed sequence of the same length as the original input. The third LSTM layer (L4) takes this compressed representation and reconstructs a sequence of the same length as the original input.

Finally, the fourth LSTM layer (L5) refines the reconstructed sequence. The output layer applies a dense transformation to each time step independently using the TimeDistributed wrapper, aiming to reconstruct the original input data. The resulting model is an LSTM autoencoder that learns to compress and reconstruct the input data while capturing temporal dependencies and patterns.

After defining a function that creates our model, we configure the training process of the LSTM autoencoder model, including the optimizer, loss function, metrics, and early stopping callback, and provide a summary of the model's architecture (Table 2).

Table 2. Summary of LSTM autoencoder model.

Layer (Type)	Output Shape	Parm
Input_1(input layer)	[(None, 1, 3)]	0
Lstm	(None, 1, 64)	17408
Lstm_1	(None, 64)	33,024
Reoeat_vector	(None, 1, 64)	0
Lstm_2	(None, 1, 32)	12,416
Lstm_3	(None, 1, 64)	24,832
Time_distributed	(None, 1, 3)	195
Total params	/	87,875
Trainable params	/	87,875
Non_trainable params	/	0

The autoencoder is compiled with the Adam optimizer, which is an efficient optimization algorithm for neural networks, and the mean squared error (MSE) is used as the loss function to measure the discrepancy between the model's predictions and the true values as it computes the average squared difference between the predicted and target values. The accuracy metric is also specified to evaluate the model's performance.

The MSE (mean square error) function is defined as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{10}$$

where n is the number of samples, y is the target value, and $\hat{y}$ is the target value.

The provided architecture performs the necessary setup and configuration for training an LSTM autoencoder model using the Adam optimizer and MSE loss function.

The early stopping is defined with twenty epochs for patience, and the model summary is finally displayed.

LSTM-AE training time per epoch is given in Figure 2. By analyzing the plot result, we notice that the model quickly converges to a relatively optimal solution. This indicates that the model has learned the underlying patterns and features of the training data efficiently within the initial epochs.

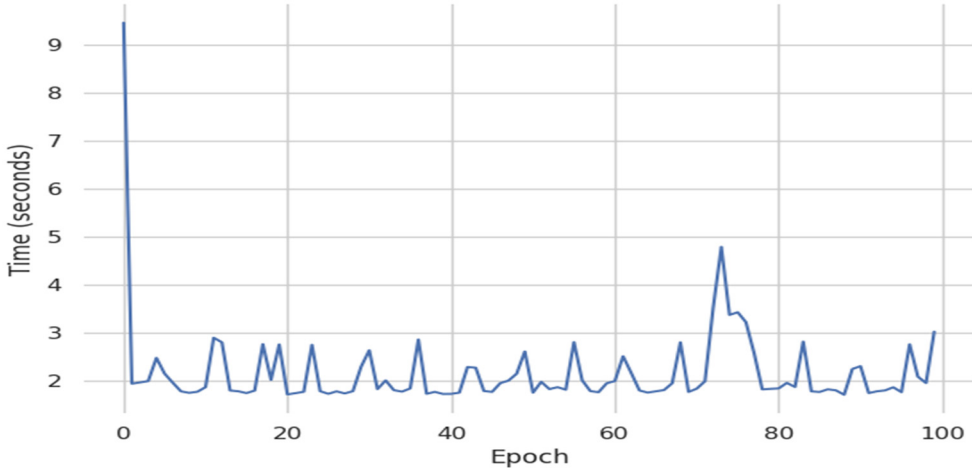


Figure 2. LSTM-AE training time per epoch.

The stagnation of training time from around the 40th epoch suggests that further training does not significantly enhance the model's performance, which is why our model saved the 65th epoch as the best one and stopped the training at the 100th epoch.

The model loss represents the difference between the reconstructed sequences (output) generated by the LSTM autoencoder model and the original input sequences (target).

The loss value is calculated using the MSE as the loss function, and the plot of the model loss over the training epochs is shown in Figure 3. It provides insights into how the loss changes as the model undergoes training.

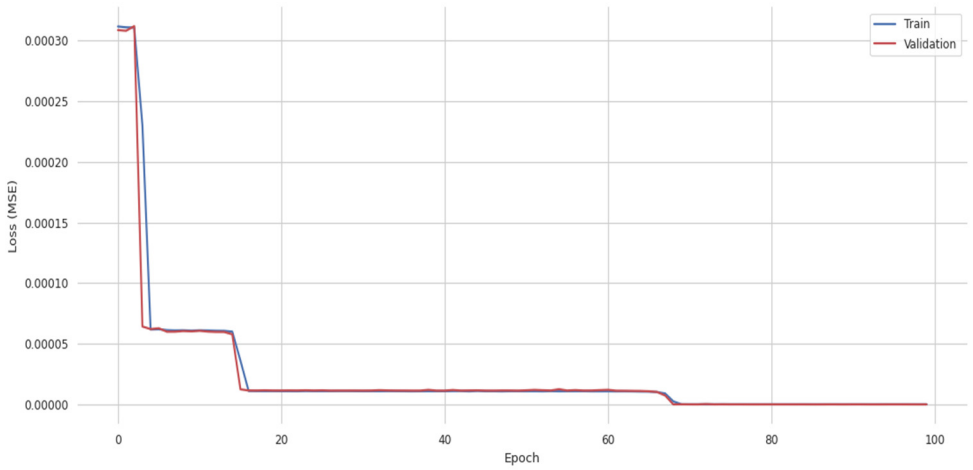


Figure 3. LSTM-AE model loss.

The plot of training accuracy and validation accuracy provides valuable insights into the model's performance during training. By comparing the two curves, we can assess the model's ability to learn and generalize.

In Figure 4, both lines increase and converge, indicating that the model is learning well and generalizing to unseen data. A large gap between the two lines may suggest overfitting, which is not our case. We can also observe the overall trend of the curves, with increasing accuracy over time indicating successful learning.

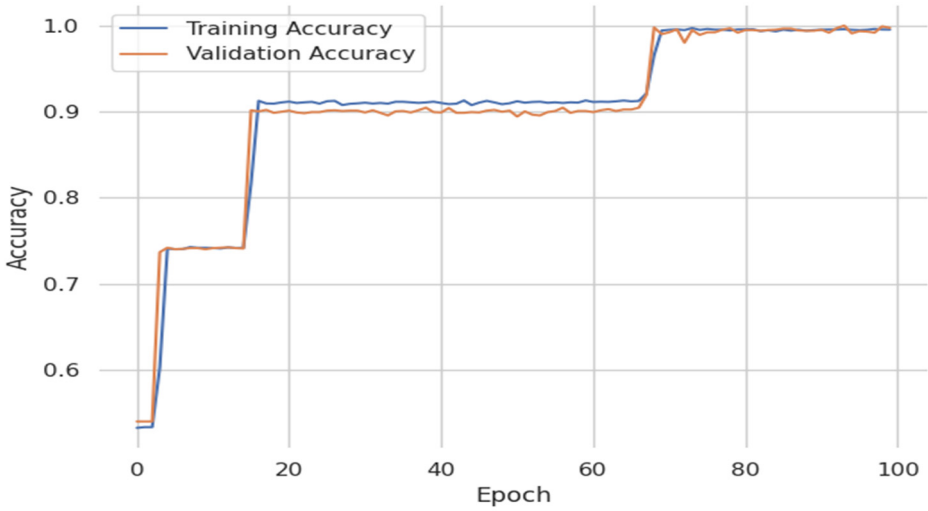


Figure 4. LSTM-AE model accuracy.

By comparing the accuracy and the loss plots, we can observe that the model's accuracy increases while the loss decreases. This indicates that the model is effectively optimizing its predictions and learning from the training data.

We conclude the LSTM-AE visualization by plotting the models' mean squared error of the imbalance axial, radial, and tangential vibrations.

The results of Figure 5 show that the MSEs of the model are very small, which means that our model learned effectively, and its performance is acceptable. The random peaks of MSE indicate the presence of anomalies which will be detected next.

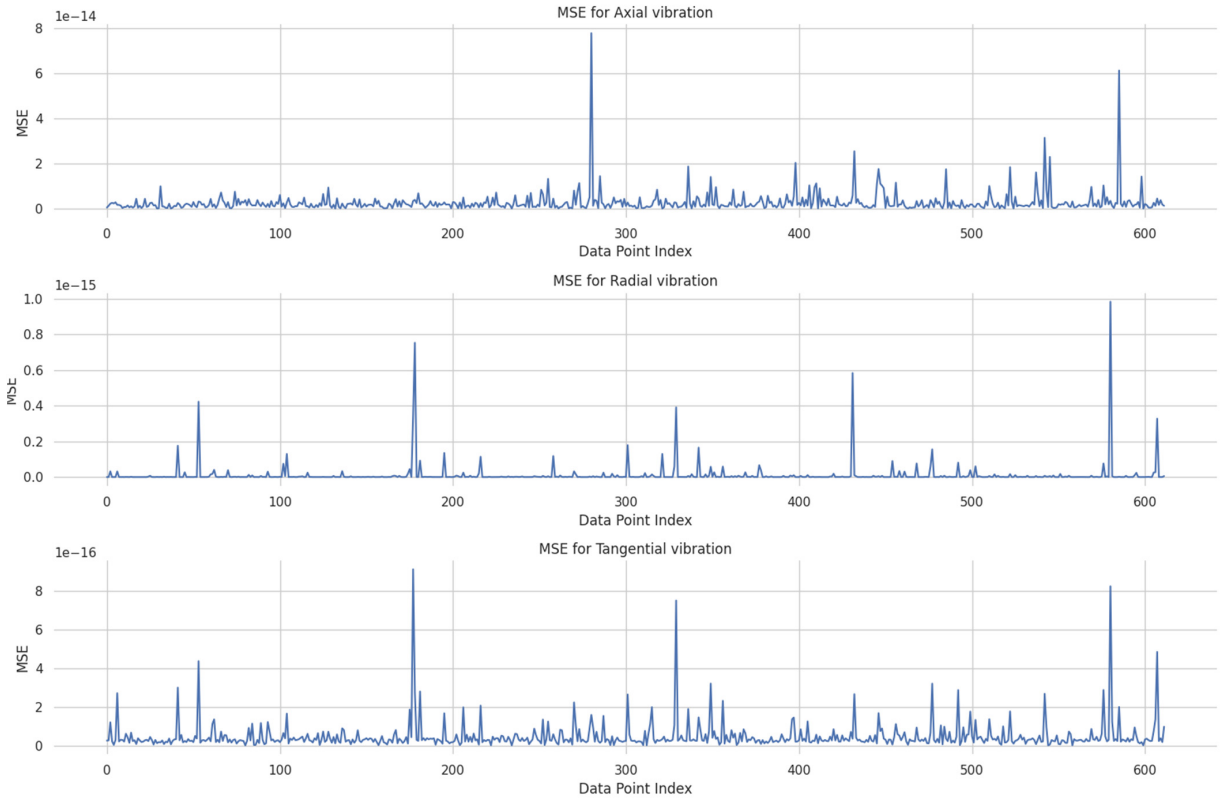


Figure 5. LSTM-AE MSE.

6. LSTM-AE Anomaly Detection

After explaining and visualizing the LSTM-AE model, we will now review its performance in detecting anomalies. Figure 6 shows 6 g MSE imbalances. We applied a 95% threshold.

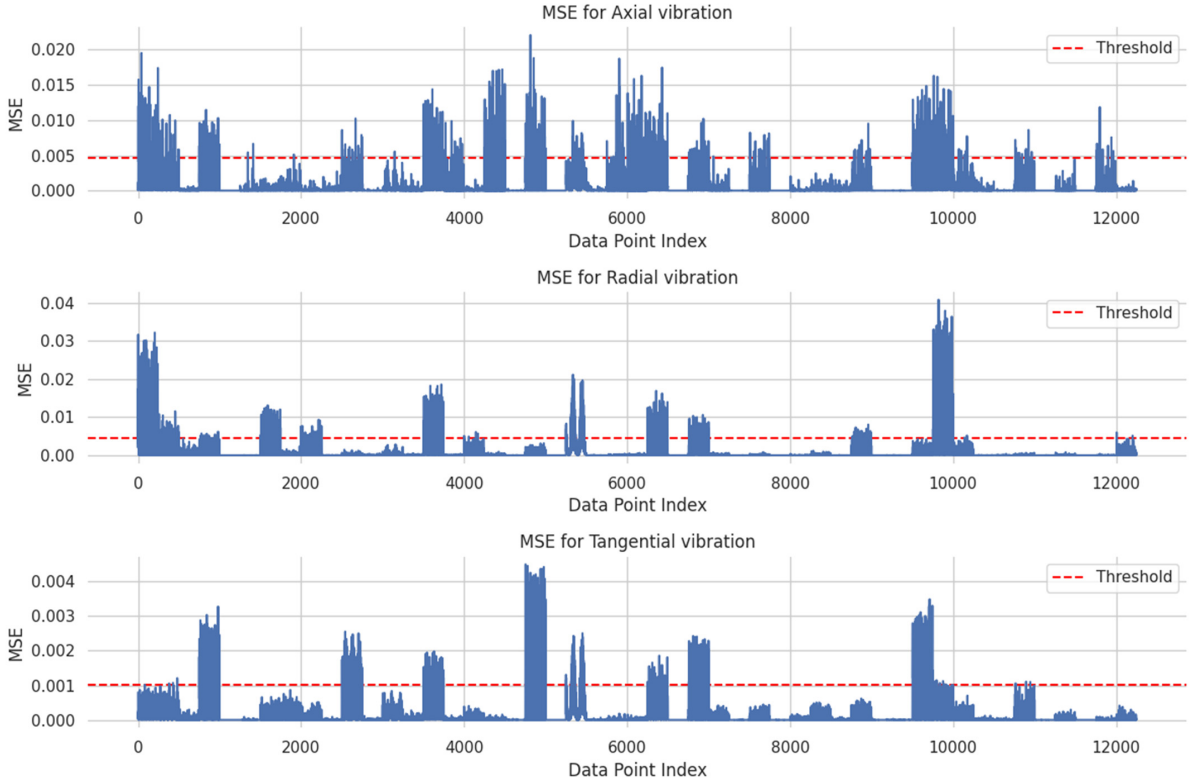


Figure 6. LSTM-AE 6 g imbalance anomalies.

Using the 95th percentile as a threshold offers a balanced approach. Higher percentiles create a more conservative threshold, reducing false positives but potentially missing some anomalies. Lower percentiles increase sensitivity to anomalies but may result in more false positives. The selection of the threshold depends on the specific application and the desired level of sensitivity and precision.

By setting the threshold at the 95th percentile, we can capture a majority of the normal data while allowing a small portion of anomalies.

7. Regular Autoencoder

Our autoencoder model is created using the standard autoencoder function with X_train as the input. The autoencoder is compiled with the Adam optimizer, which is an efficient optimization algorithm for neural networks, and the mean squared error (MSE) is used as the loss function to measure the discrepancy between the model's predictions and the true values as it computes the average squared difference between the predicted and target values. The accuracy metric is also specified to evaluate the model's performance.

The summary of the autoencoder model shows the architecture and the number of parameters (Table 3). The model consists of four layers:

Table 3. Summary of the AE model.

Layer (Type)	Output Shape	Param
Input_1(input layer)	[(None, 1, 3)]	0
Dense	(None, 1, 128)	512
Dense_1	(None, 1, 128)	16,512
Dropout	(None, 1, 128)	0
Dense_2	(None, 1, 3)	387
Total params	/	17,411
Trainable params	/	17,411
Non_trainable params	/	0

Then, we plot the epochs' training time, which allows us to identify any significant variations or trends in training time and provide insights into the efficiency of the training process.

Figure 7 shows that the training time decreases significantly during the first epochs from 2.19 s to 0.91 s, and fluctuates from around the 9th epoch until the last epoch 1.41 s to 0.7 s. This means that our model reached its optimal capacities in the first epochs, and continuing the training will not result in major improvements.

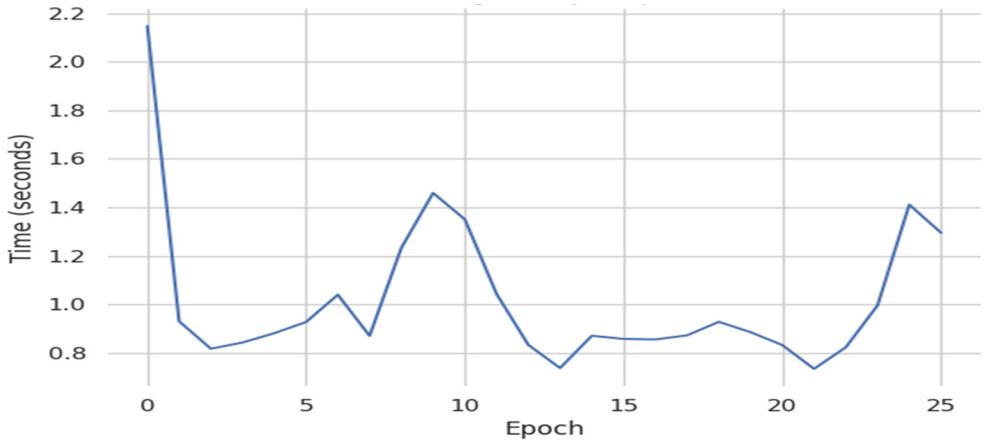


Figure 7. AE training time per epoch.

The model loss plot is given by the Figure 8:

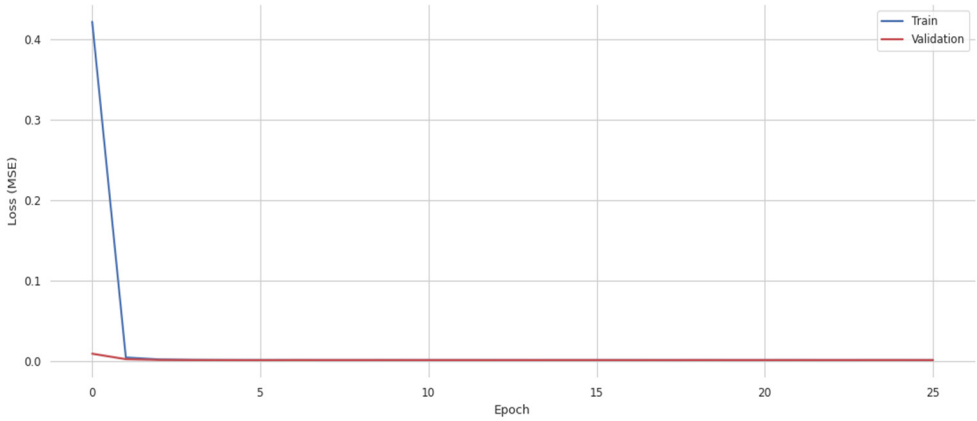


Figure 8. Autoencoder model loss.

We can see that the training loss decreases and reaches stability within the first epochs, overlapping with the validation loss. It means that the model has learned the underlying patterns and can make accurate predictions on both the training and validation datasets. It quickly adapts to the data and reduces the loss, reaching its full potential early on, suggesting that continuing the training process may only result in minor improvements.

We conclude by visualizing the MSE for each vibration (axial, radial, and tangential) in the predicted output compared to the original test data. See Figure 9.

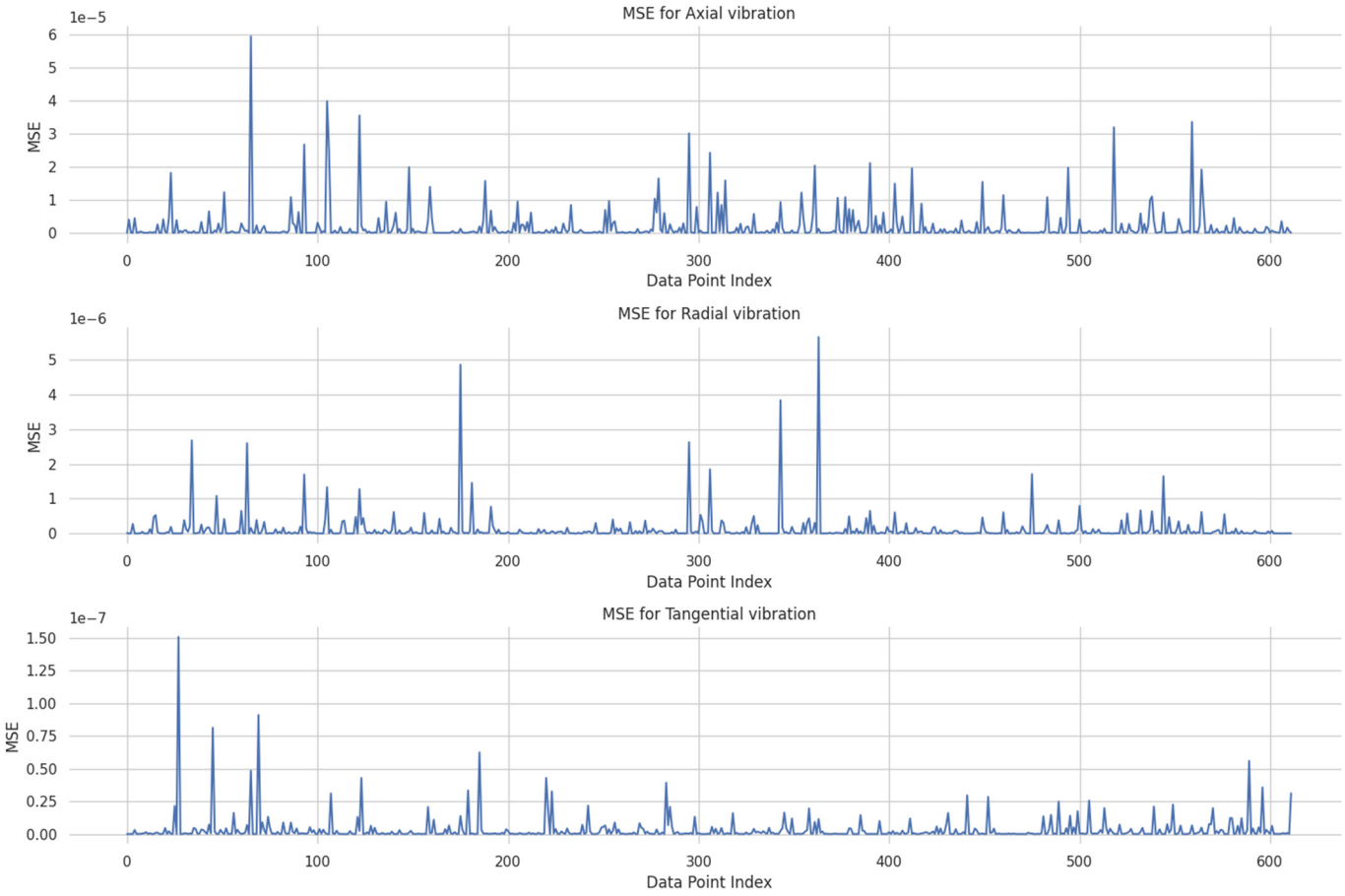


Figure 9. Autoencoder MSE.

The MSE plotted in Figure 9 assesses how well the autoencoder model is reconstructing each feature. The MSE values are very small (from 10^{-5} to 10^{-7}), which indicates that the

predicted values are closer to the original values, implying better reconstruction accuracy and overall performance.

Finally, the anomaly detection step of the 6 g imbalance data is given in Figure 10:

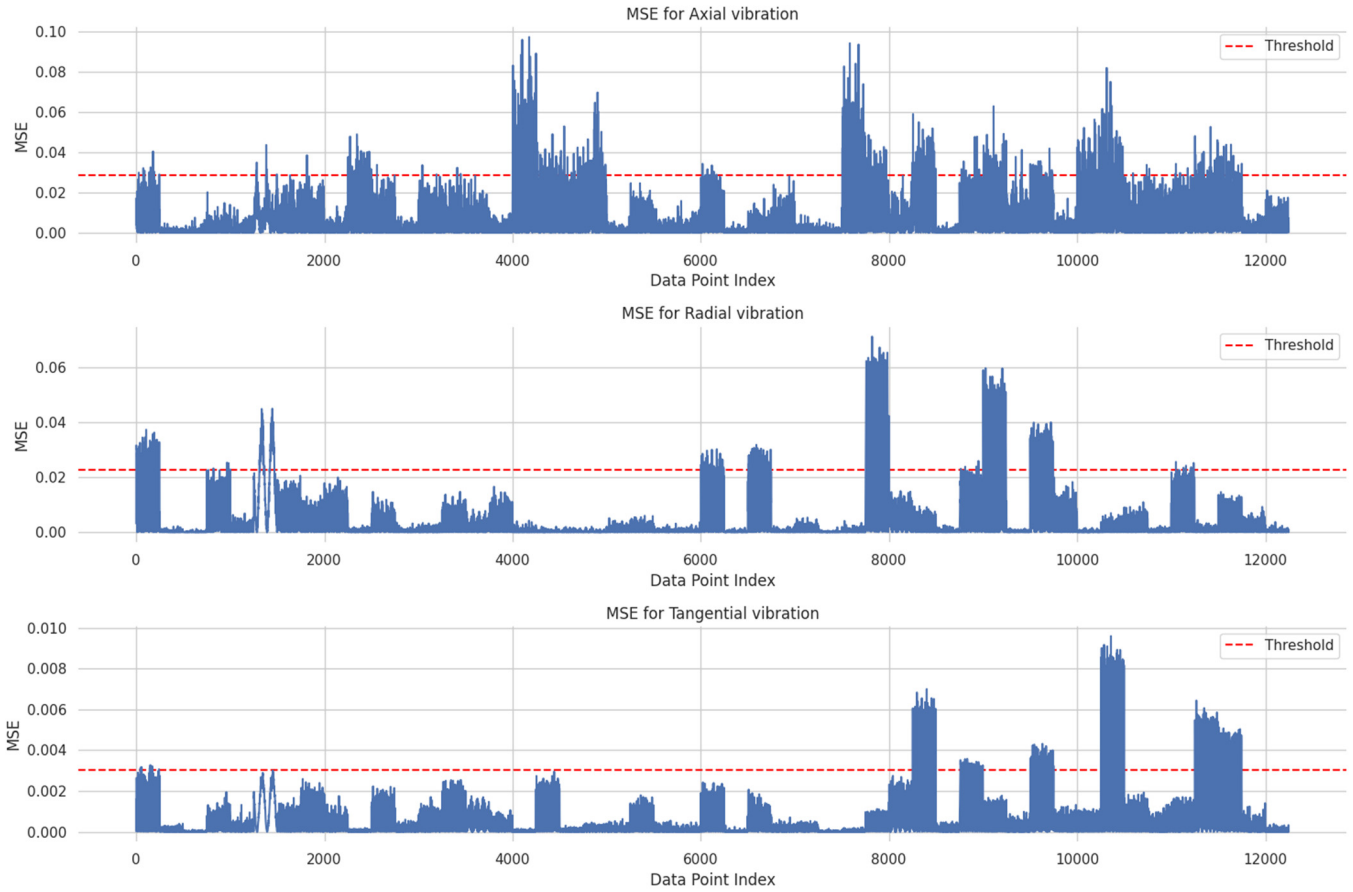


Figure 10. AE 6 g imbalance anomalies.

8. LSTM-AE and Regular AE Comparison

After introducing both models, we will now compare their performance by reviewing three aspects.

8.1. Training Time

After plotting the training time of both models on the same data, we notice that the training process of the LSTM-AE took a significant amount of time compared to the training process of the regular AE (6 min vs. 40 s).

This can be explained by the more complex architecture of the LSTM-AE. The LSTM-AE requires more time for each epoch due to the additional computations involved in training the LSTM layers. These computations include the forward and backward propagation of information through the recurrent connections and updating of the LSTM cell states.

Consequently, the overall training process takes longer compared to the regular AE, which has a simpler architecture and fewer computational operations.

8.2. Loss Functions

The MSE loss functions of both models decrease significantly with time and reach a plateau, suggesting that both models successfully learned the data patterns and reached their optimal performance.

On the other hand, the MSE loss values of the LSTM-AE were remarkably less important than the loss values of the regular AE (0.4 vs. 0.0003), Figures 3 and 8, which proves

the superiority of the LSTM-AE in handling large, complex amounts of data and detecting temporal features and dependencies.

The LSTM layers allow the model to learn and exploit the temporal relationships between the input features. This enables the LSTM-AE to better reconstruct the input data and minimize the reconstruction error, as quantified by the MSE loss function. In contrast, the regular AE lacks the ability to explicitly model and capture temporal dependencies. It treats the input data as independent and identically distributed samples, neglecting any underlying sequential information. As a result, the regular AE may struggle to effectively reconstruct the time-dependent patterns in the data, leading to higher MSE loss values.

By leveraging the memory cells and recurrent connections, the LSTM-AE is able to better preserve the temporal information and reconstruct the input data with higher fidelity, resulting in lower MSE loss values. This highlights the advantage of using LSTM-based architectures when dealing with sequential or time-dependent data.

8.3. MSE Anomalies

While both models had impressive low mean squared error values on the different axes of the weighted imbalances, the MSEs of the LSTM-AE were significantly smaller than the ones on the regular AE (10^{-7} vs. 10^{-16}). Figures 5 and 9.

This means that while both models are performing well, the LSTM-AE performances are superior due to its complexity and capability to handle big amounts of data and temporal dependencies, which is confirmed by the loss results discussed previously.

The remarkably smaller values of the MSE in the LSTM-AE also indicates that the anomalies present and detected in the machinery are less important and less common compared to the anomalies in the regular AE model, which only consolidate and confirm our results and findings.

9. Conclusions

Artificial intelligence plays a significant role in the field of anomaly detection. The proposed work focuses on a combination of the two architectures: LSTM layers were added to the autoencoder in order to leverage the LSTM capacity for handling large amounts of temporal data. After developing the LSTM-AE model, its performances are compared to the regular AE model of an electrical motor. To prove the efficiency of the model, first, the regular autoencoder is introduced and both models are trained on the same data using the same code written with Python. After visualizing the results and competence of the two models, a comparison of their performance on three points, training time, loss function, and MSE anomalies, was given. The analysis performed clearly shows that the LSTM-autoencoder had significantly smaller loss values (0.0003 vs. 0.4) and MSE anomalies (10^{-16} vs. 10^{-7}) compared to the regular autoencoder, while the regular autoencoder outperformed the LSTM when it came to training time (40 s vs. 6 min). Lastly, the LSTM-autoencoder had superior performance although it was slower than the regular autoencoder due to the complexity of the LSTM layers added.

Finally, the choice between the two models depends on the specific requirements of the application, weighing the trade-off between training time and performance. The most appropriate DL model or approach may vary depending on the systems' characteristics, specific requirements, data features, and the goals of the PdM application.

The real-time monitoring integration and feedback mechanisms, and a comparison of the LSTM-autoencoder with other methods such as generative models, variational autoencoder (VAE), and temporal logic-based learning could be examined in future research.

Author Contributions: Conceptualization, F.L. and M.B.; methodology, F.L.; software, F.L., A.B. and H.H.; validation, F.L., S.A.T., A.B. and H.H.; formal analysis, F.L.; investigation, F.L. and S.A.T.; resources, A.B. and H.H.; data curation, F.L.; writing—original draft preparation, F.L. and S.A.T.; writing—review and editing, F.L. and M.B.; visualization, F.L. and S.A.T.; supervision, F.L.; project administration, F.L., S.A.T. and M.B.; funding acquisition, F.L. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Directorate General for Scientific Research and Technological Development DGRSDT, Ministry of Higher Education and Scientific Research, Algeria.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Coandă, P.; Avram, M.; Constantin, V. A state of the art of predictive maintenance techniques. In Proceedings of the 9th International Conference on Advanced Concepts in Mechanical Engineerin, Iași, Romania, 4–5 June 2020. [CrossRef]
2. Wentz, V.H.; Maciel, J.N.; Gimenez Ledesma, J.J.; Ando Junior, O.H. Solar Irradiance Forecasting to Short-Term PV Power: Accuracy Comparison of ANN and LSTM Models. *Energies* **2022**, *15*, 2457. [CrossRef]
3. Dong, S.; Wang, P.; Abbas, K. A survey on deep learning and its applications. *Comput. Sci. Rev.* **2021**, *40*, 100379. [CrossRef]
4. Böhm, L.; Kolb, S.; Plankenbühler, T.; Miederer, J.; Markthaler, S.; Karl, J. Short-Term Natural Gas and Carbon Price Forecasting Using Artificial Neural Networks. *Energies* **2023**, *16*, 6643. [CrossRef]
5. Son, N.; Yang, S.; Na, J. Hybrid Forecasting Model for Short-Term Wind Power Prediction Using Modified Long Short-Term Memory. *Energies* **2019**, *12*, 3901. [CrossRef]
6. Basora, L.; Olive, X.; Dubot, T. Recent Advances in Anomaly Detection Methods Applied to Aviation. *Aerospace* **2019**, *6*, 117. [CrossRef]
7. Son, N.; Jung, M. Analysis of Meteorological Factor Multivariate Models for Medium- and Long-Term Photovoltaic Solar Power Forecasting Using Long Short-Term Memory. *Appl. Sci.* **2021**, *11*, 316. [CrossRef]
8. Peres, R.S.; Jia, X.; Lee, J.; Sun, K.; Colombo, A.W.; Barata, J. Industrial Artificial Intelligence in Industry 4.0-Systematic Review. *Chall. Outlook IEEE Access* **2020**, *8*, 220121–220139. [CrossRef]
9. Zonta, T.; Da Costa, C.A.; da Rosa Righi, R.; de Lima, M.J.; da Trindade, E.S.; Li, G. Predictive maintenance in the industry 4.0: A systematic literature review. *Comput. Ind. Eng.* **2020**, *150*, 106889. [CrossRef]
10. Zhang, C.; Lu, Y. Study on artificial intelligence: The state of the art and future prospects. *J. Ind. Inf. Integr.* **2021**, *23*, 100224. [CrossRef]
11. Wan, J.; Tang, S.; Li, D.; Wang, S.; Liu, C.; Abbas, H.; Vasilakos, A.V. A Manufacturing Big Data Solution for Active Preventive Maintenance. *IEEE Trans. Ind. Inform.* **2017**, *13*, 2039–2047. [CrossRef]
12. Poór, P.; Basl, J.; Zenisek, D. Predictive Maintenance 4.0 as next evolution step in industrial maintenance development. In Proceedings of the 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 28 March 2019; pp. 245–253. [CrossRef]
13. Rzepka, F.; Hematty, P.; Schmitz, M.; Kowal, J. Neural Network Architecture for Determining the Aging of Stationary Storage Systems in Smart Grids. *Energies* **2023**, *16*, 6103. [CrossRef]
14. Çınar, Z.M.; Abdussalam Nuhu, A.; Zeeshan, Q.; Korhan, O.; Asmael, M.; Safaei, B. Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0. *Sustainability* **2020**, *12*, 8211. [CrossRef]
15. Theissler, A.; Pérez-Velázquez, J.; Kettelgerdes, M.; Elger, G. Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. *Reliab. Eng. Syst. Saf.* **2021**, *215*, 107864. [CrossRef]
16. Serradilla, O.; Zugasti, E.; Rodriguez, J.; Zurutuza, U. Deep learning models for predictive maintenance: A survey, comparison, challenges and prospects. *Appl. Intell.* **2022**, *52*, 10934–10964. [CrossRef]
17. Janiesch, C.; Zschech, P.; Heinrich, K. Machine learning and deep learning. *Electron. Mark.* **2021**, *31*, 685–695. [CrossRef]
18. Zhou, Z.H. A brief introduction to weakly supervised learning. *Natl. Sci. Rev.* **2018**, *5*, 44–53. [CrossRef]
19. Na Pattalung, T.; Ingviya, T.; Chaichulee, S. Feature explanations in recurrent neural networks for predicting risk of mortality in intensive care patients. *J. Pers. Med.* **2021**, *11*, 934. [CrossRef] [PubMed]
20. Van Engelen, J.E.; Hoos, H.H. A survey on semi-supervised learning. *Mach. Learn.* **2020**, *109*, 373–440. [CrossRef]
21. Dike, H.U.; Zhou, Y.; Deveerasetty, K.K.; Wu, Q. Unsupervised learning based on artificial neural network: A review. In Proceedings of the 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS), Shenzhen, China, 25–27 October 2018; pp. 322–327. [CrossRef]
22. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An introduction to deep reinforcement learning. *Found. Trends Mach. Learn.* **2018**, *11*, 219–354. [CrossRef]
23. Kiranyaz, S.; Avci, O.; Abdeljaber, O.; Ince, T.; Gabbouj, M.; Inman, D.J. 1D convolutional neural networks and applications: A survey. *Mech. Syst. Signal Process.* **2021**, *151*, 107398. [CrossRef]
24. Salehinejad, H.; Sankar, S.; Barfett, J.; Colak, E.; Valaee, S. Recent advances in recurrent neural networks. *arXiv* **2017**, arXiv:1801.01078. [CrossRef]
25. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef] [PubMed]
26. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]

27. Okut, H. Deep learning for subtyping and prediction of diseases: Long-short term memory. In *Deep Learning Applications*; IntechOpen: London, UK, 2021. [CrossRef]
28. Berrajaa, A. Natural Language Processing for the Analysis Sentiment using a LSTM Model. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 777–785. [CrossRef]
29. Lindemann, B.; Maschler, B.; Sahlab, N.; Weyrich, M. A survey on anomaly detection for technical systems using LSTM networks. *Comput. Ind.* **2021**, *131*, 103498. [CrossRef]
30. Bank, D.; Koenigstein, N.; Giryes, R. Autoencoders. In *Machine Learning for Data Science Handbook*; Springer: Cham, Switzerland, 2020. [CrossRef]
31. Saumya, S.; Singh, J.P. Spam review detection using LSTM autoencoder: An unsupervised approach. *Electron. Commer. Res.* **2022**, *22*, 113–133. [CrossRef]
32. Do, J.S.; Kareem, A.B.; Hur, J.W. LSTM-Autoencoder for Vibration Anomaly Detection in Vertical Carousel Storage and Retrieval System (VCSRS). *Sensors* **2023**, *23*, 1009. [CrossRef]
33. Nguyen, H.D.; Tran, K.P.; Thomassey, S.; Hamad, M. Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management. *Int. J. Inf. Manag.* **2021**, *57*, 102282. [CrossRef]
34. Smagulova, K.; James, A.P. A survey on LSTM memristive neural network architectures and applications. *Eur. Phys. J. Spec. Top.* **2019**, *228*, 2313–2324. [CrossRef]
35. Pulver, A.; Lyu, S. LSTM with working memory. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 845–851. [CrossRef]
36. Sublime, J.; Cabanes, G.; Matei, B. Study on the Influence of Diversity and Quality in Entropy Based Collaborative Clustering. *Entropy* **2019**, *21*, 951. [CrossRef]
37. Malhotra, P.; Vig, L.; Shroff, G.; Agarwal, P. Long Short-Term Memory Networks for Anomaly Detection in Time Series. In Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 22–24 April 2015; p. 89. Available online: http://www.i6doc.com/en/ (accessed on 19 January 2017).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.