ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

Anomaly Detection System for Stepper Motors

[1] Labib Sharrar

[1] Universiti Teknologi Malaysia.
Corresponding Author Email: [1] sharrar@graduate.utm.my

Abstract*— Predictive maintenance (PdM) systems have the potential to autonomously detect underlying motor issues at early stages. Although many such systems have been proposed up to data, they have yet to be implemented. Most of these methods, which are based on supervised learning, require hours of manual data collection and annotation. Furthermore, they are mostly made to tackle a single instead of the multiple motor issues that may occur and are unable to adapt to varying motor speed and load conditions. Thus, they are not viable for industrial implementation. Therefore, this paper presents an unsupervised LSTM autoencoder-based anomaly detection system for electric motors. It analyzes the vibration and current consumption data from motors to detect anomalies, which is sufficient to account for the various motor defects. The system comes with a variety of features that allows users to autonomously collect data, train models and deploy models. In addition to that, users can remotely keep track of the motor's conditions. To test the system, a hardware test bench using a stepper motor is made to simulate defective conditions. The LSTM Autoencoder-based anomaly detection system is described step-by-step in this paper.

Index Terms* — Predictive Maintenance, Fault Diagnosis, Electric Motors, Long-Short Term Memory, Autoencoders.

I. INTRODUCTION

Electric motors are crucial components in modern manufacturing industries. They can be found in robot arms, automated guided vehicles, power generators, conveyor belts, etc. Currently, due to high manufacturing demand from users, electric motors operate continuously for 24 hours every day. This can cause motors to develop a multitude of defects, which if not treated in time, could lead to machine breakdown, production downtime, and financial losses [1]. Therefore, industries are increasingly looking into predictive maintenance (PdM) systems for motors to overcome such issues. Unlike conventional maintenance strategies, PdM systems use an array of sensors to help collect real-time data and use it to develop models or algorithms that can detect underlying issues in machines [2]. PdM systems are both less exhaustive and less prone-to-errors, unlike their traditional counterparts [3].

The process of fault diagnosis of motors (FDM) happens to be a branch of PdM. The literature is filled with potential FDM systems [4]. Unfortunately, the majority of the FDM systems proposed by researchers make use of supervised learning models [5], [6], [7] and they suffer from certain flaws. In the case of supervised systems, manual data collection is required. After that, the data has to be annotated before training the model. This can be time-consuming for companies. Furthermore, most of the proposed systems were meant to tackle a single type of motor fault, whereas multiple faults may occur in an electric motor, such as bearing fault, rotor fault, stator issues, etc. Annotating the data for each of these faults would not only require time but would also require manpower as experts would be needed to interpret the features and annotate [8]. On top of that, supervised methods require users to simulate motor faults to collect the necessary data, so that fault classification can be made. This is a great challenge since accurate modeling of each fault is difficult [9].

Additionally, separate models have to be developed depending on the speed or load carried by electric motors. This means that models developed for a particular motor may not be applicable all the time, since the same type of motor in different machines may operate with varying speeds and load conditions. Overall, there are too many challenges to overcome before implementing an FDM system for motors. Thus, FDMs remain elusive in industries. Most of the proposed solutions have proven to not be viable for companies.

What many people often overlook is that rather than developing complex supervised models, underlying issues can be detected by simply identifying anomalies [11]. Companies do not care what type of fault is occurring in their motor, they simply want to know whether the motor is working well. As such, detecting anomalies in real-time in electric motors can be sufficiently helpful. Unlike most FDM techniques, anomaly detection can be an unsupervised process and may not require manually annotated datasets [12]. Thus, this paper presents a long-short term memory (LSTM) autoencoder-based anomaly detection system for electric motors. The presented system is nearly unsupervised. Once connected to a motor, it can autonomously collect the vibration and current consumption data from the motor, train the LSTM autoencoder model and then deploy the model. The model analyzes the vibration and current signals from the motor to look for anomalies. The main part of the system comes in the form of a PC user interface that can easily be operated by a user. It also sends notifications to users. A web user interface (UI) is also part of the system, which allows users to observe the trend in data. The working process of the presented system is explained in detail in the next section.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

II. METHODOLOGY

A. System Overview

The presented system is unsupervised. It consists of INA219 current and MPU6050 accelerometer sensors, which are for measuring the current consumption and vibration of the motor respectively. The sensors are interfaced with an Arduino Uno board is serially connected to a PC where the main algorithm for the presented system runs. The main system is in the form of a UI that enables users to initiate autonomous data collection, model training, and model deployment. It will collect data for a week and then use it to train an LSTM autoencoder model. After training the model is deployed. This model will then look for data points that deviate from the regular pattern. Using the model, the system will calculate the rate of anomalies occurring per second. If the rate of anomalies per second exceeds a threshold of 0 anomalies per second (a/s) the system will send an SMS or email notification to the user as a warning. The notification message to the user will contain a link that can enable the user to access a web UI. Through this, it is possible to observe the trends in vibration and current consumption data, as well as the overall status of the motor. A block diagram representing the overall system is shown in Fig. 1.

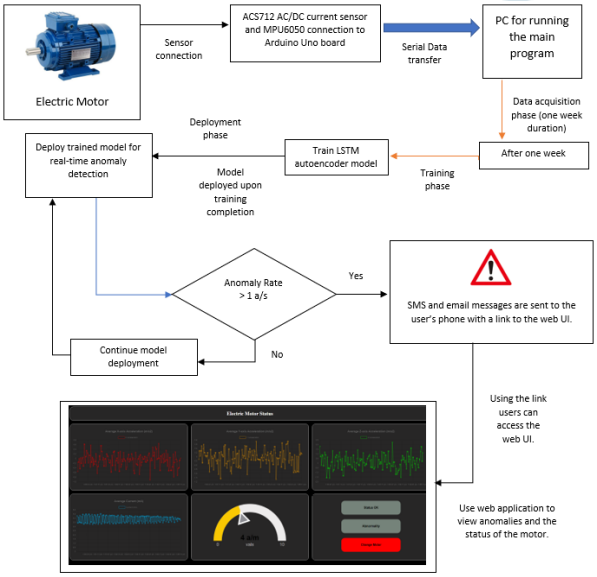


Fig. 1. Overview of the motor anomaly detection system.

B. Experimental Setup

To test the anomaly detection system, an experimental setup is made. It is used to simulate a motor bearing defect, so that we may understand whether the presented system can truly detect abnormalities. The setup is a slider that is controlled by a Nema17 stepper motor. It has a custom-made control box which is also run by an Arduino. A labeled image of the experimental setup used to test the system is shown in Fig. 2. The objective of this research is to build a motor anomaly detection system that can apply to motors in different machines. The stepper motor in the setup represents the motor that the system will be deployed on to look for anomalies, while the rest of the slider represents the machine parts controlled by the motor.

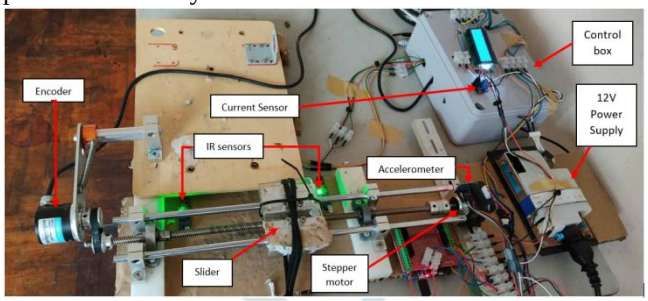


Fig. 2. The experimental hardware setup to simulate motor bearing defect.

As shown in the image above, the experimental setup has various sensors. The infrared (IR) sensors act as limit switches for the slider. The accelerometer sensor is placed on top of the stepper motor to measure the vibration, while the current sensor is connected to the motor's power supply. It should be noted that the slider has a PID controller, which ensures that the motor maintains the same speed even when additional load is placed on the slider. This is done to make the bearing fault simulation more realistic.

Now the question is, how can this hardware setup be used to simulate a bearing defect? A bearing is a delicate mechanical component that holds the shaft of a motor in place. If a bearing degrades, the motor shaft has trouble moving. This can cause the motor's current consumption as well as vibration to change. In the works of Chopade et al. [13], Ozcan et al. [14], and Shao et al. [15], bearing defects were simulated by placing loads on the motor shaft, so that there is stress on the bearing. This made it higher for the shafts to move and caused changes in current consumption as well as vibration. Similar to the work of these previous researchers, loads were also added to this experimental setup to simulate a bearing defect. This was done by typing a basket to the slider and placing weights in the basket. An image of how the weights are added is shown in Fig. 3.

Fig. 3. Basket tied to the slider (left) where the weights (right) are placed.

ISSN (Online) 2394-6849

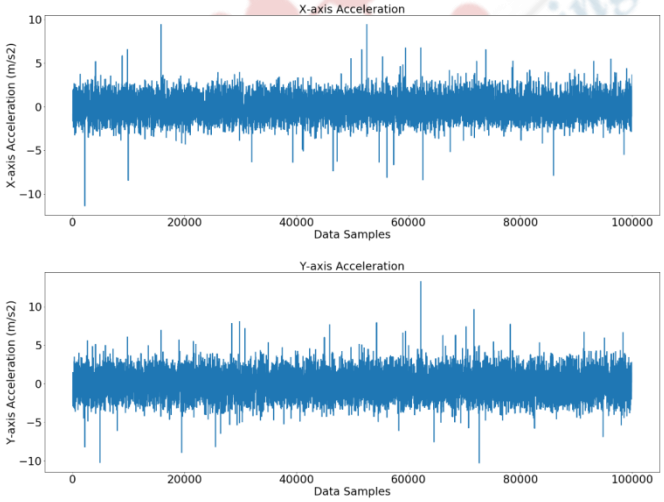
International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

The weights, which are dumbbells of 1.25kg each, are tied to the slider from the side to produce unbalanced conditions on the motor. This would place more stress on the motor bearing. So basically, when the slider is operating without any weights in the basket, it is considered to be operating under normal conditions. As the dumbbells are placed into the basket one after the other, conditions on the motor become more unbalanced and the bearing defect is considered to become more severe as the motor shaft has more difficulty moving. The anomaly detection system is deployed in real-time on the setup to detect abnormalities as the load added increases. The system is tested with different load conditions as well as motor speed. Data analysis from this setup is explained in later sections.

C. Data Analysis

Before the overall system can be explained, it was necessary to understand trends in data from the stepper motor, especially as the weights are added to the slider. It was important to know whether the experimental setup could truly be used to test the presented system. We needed to know if an unbalanced load caused variations in vibration and current consumption like a bearing defect would. It should be clarified that vibration (acceleration along the x-axis, y-axis, and z-axis) and current consumption data are collected from the stepper motor described in the earlier sub-section. The unit of acceleration along the three axes is m/s^2, while for current consumption it is ampere (A). Specifically for the purpose of data analysis, the stepper motor is run at 200 RPM and a weight of 2.5kg is added to it to simulate a bearing defect. The trend in data from the motor while running at 200 RPM is shown in Fig 4. After collecting data under normal conditions, a weight of 2.5kg was placed in the slider's basket. Data for this instance was collected to see if the unbalanced conditions caused changes in the data patterns. The data trend under this condition is shown in Fig 5.



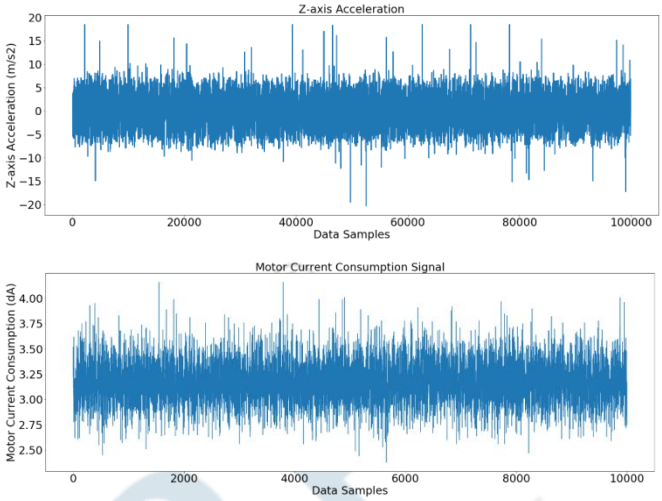


Fig 4: The trend in motor vibration and current consumption data.

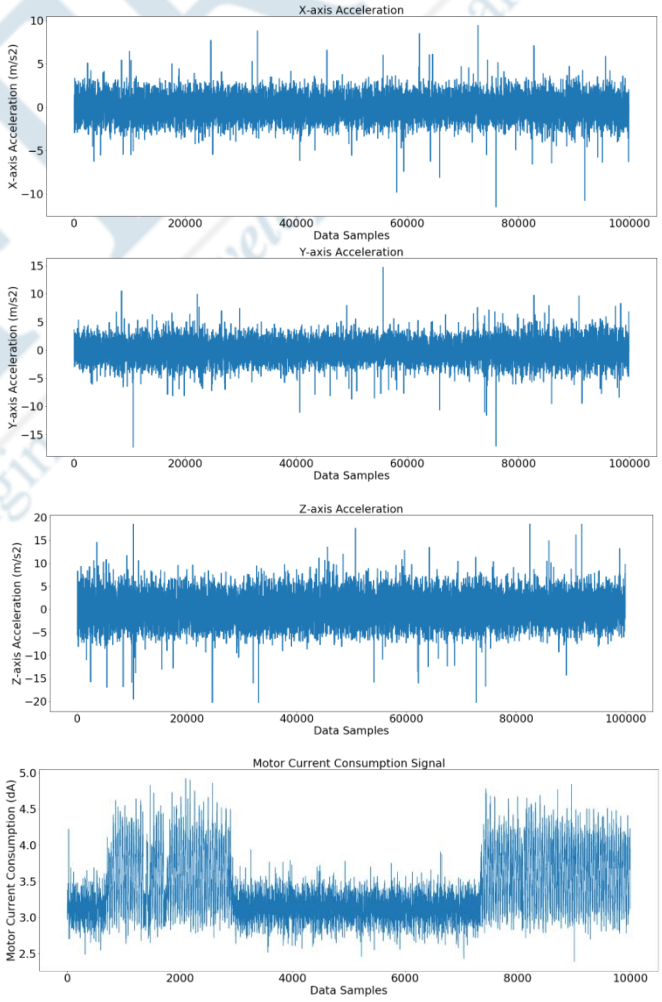


Fig 5. Trend in motor data after the 2.5kg weights are added to the slider.

By comparing the graphs in Fig. 4 and Fig. 5, we can see that for the case of the Nema17 stepper motor, there seems to

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

be little change among the x-axis, y-axis, and z-axis data. However, we can see a lot of change in terms of current consumption. The current consumption pattern of the stepper motor changes as weights are added to the slider's basket. Due to the unbalanced conditions, the current consumption of the stepper motors changes just as it would have if there was an actual bearing defect. An LSTM autoencoder model can easily be trained to analyze the current data pattern and detect abnormalities.

D. LSTM Autoencoder Model

The LSTM is a family of neural networks that can be used to analyze continuous data. It is a variation of the recurrent neural networks that were made to solve the issue of vanishing gradient [16]. Autoencoders on the other hand are neural networks that are applied for unsupervised learning. They can be used to efficiently learn data patterns and ignore the noise [17]. Autoencoders can be used to check whether data deviates from the regular pattern. Typical autoencoder models consist of encoder and decoder layers. The encoder compresses the input signals to latent features, while the decoder decompresses those features and reconstructs the input signals [18]. Autoencoders are trained on normal data patterns. Therefore, they reconstruct the signals based on familiar signal patterns. During deployment, the outputs and inputs of the autoencoder model can be compared to check data patterns and identify possible anomalies [19]. Since the vibration and current consumption data from the stepper motor are continuous and a machine learning model is needed to accurately learn the patterns in data, an LSTM autoencoder model is chosen for the motor anomaly detection system. This type of model is also chosen due to its effectiveness for motor fault diagnosis as shown in the works of Principi et al. [20] and Huang et al. [21]. The structure of the LSTM autoencoder model used for this project is shown in Fig. 6. It should be noted that this model is entirely custom-made.

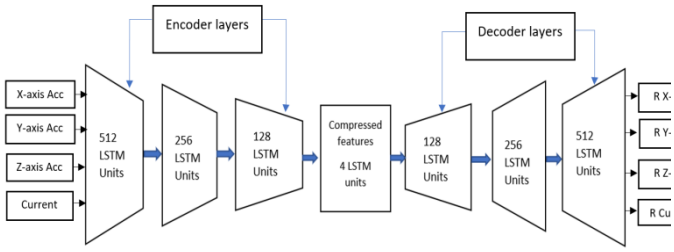


Fig. 6. Structure of the LSTM autoencoder model.

As shown in the image, the encoder layer consists of three hidden layers containing 512, 256, and 128 units respectively. The decoder layer consists of the same number of layers and units but with reverse connections. The middle layer consists of four units that hold the compressed features of the input data[22]. After the encoder compresses the input signals, the decoder reconstructs the input signals from the compressed features. Then the error between the original and reconstructed signals is calculated. How this error calculation helps to segregate anomalies is explained in the next sub-section. Examples of original and reconstructed signals are shown by Fig. 7 and Fig. 8.

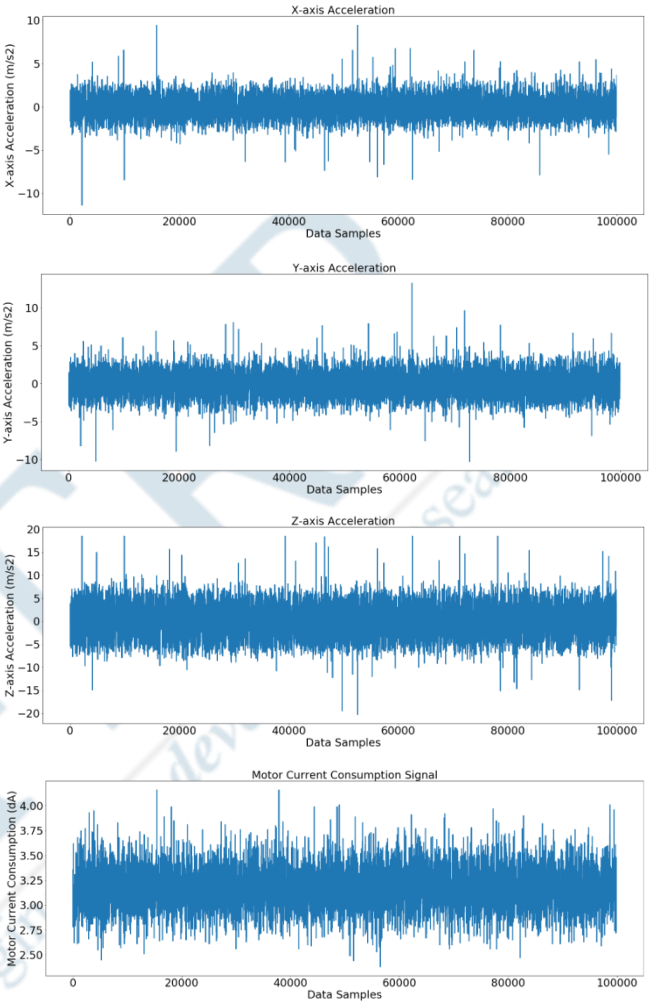
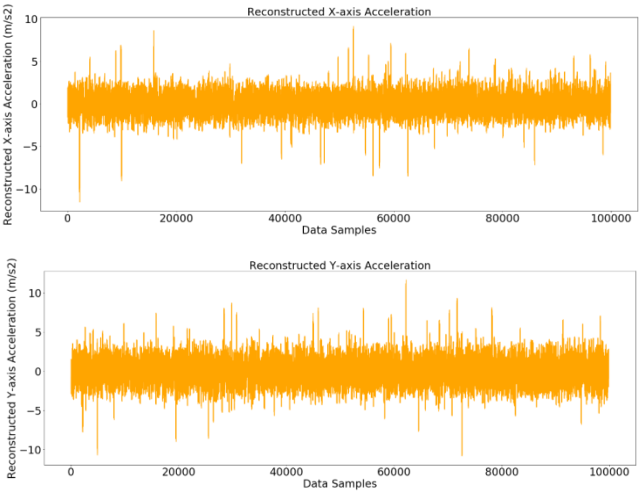


Fig. 7. The original stepper motor signals.



ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

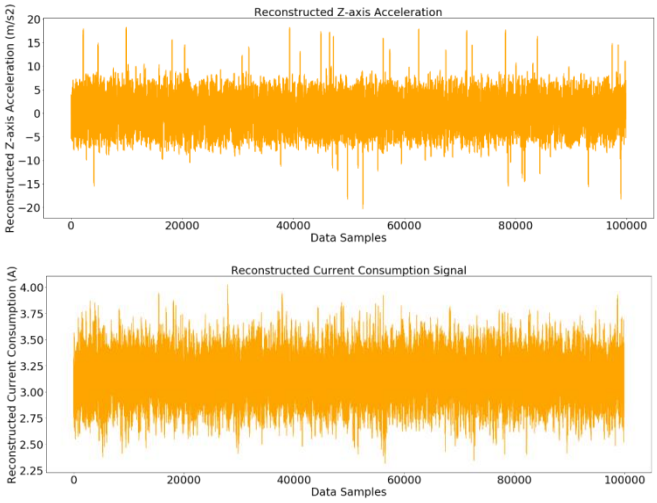


Fig. 8. The reconstructed stepper motor signals by the LSTM autoencoder model.

This LSTM autoencoder model is trained for 10-time steps (a sequence of 10 samples at each instant), with a batch size of 10 and 200 epochs. The dataset which is used to train the model is split into training and validation datasets with a ratio of 70 to 30 respectively. Only models with decremental validation losses are saved. Validation loss is chosen for performance measurement because it is a regression, and not a classification, problem. Thus, the final model saved each time the LSTM autoencoder model is trained has the lowest possible validation loss.

E. Anomaly Segregation by Calculating Mean Square Error and Thresholding

Now we know that LSTM autoencoders can reconstruct familiar signals, and anomalies can be found by calculating errors between original and reconstructed signals. However, how is this calculation done? If the error difference between a certain sample of the original and the reconstructed signal is above a certain threshold, then that sample will be classified as an anomaly. In this project, the error difference between the original and reconstructed signals is calculated using mean square error (MSE). So, if anomalous data is fed into the model, the MSE between its original and reconstructed signals should be higher than when the normal data is used. Overall, if the MSE of a data sample is above a certain threshold, it is classified as an anomaly. To determine the MSE threshold, the LSTM autoencoder model is deployed on the data that is used to train it (training data is considered as normal by the LSTM autoencoder model). As the model is deployed, the MSE between samples of the original and reconstructed signals is calculated.

To understand the overall anomaly segregation process, let us go through a more elaborate example. Before this, it should be noted that the LSTM autoencoder model is trained with a timestep of 10. That means at each step, 10 samples of each variable are fed into the model. For example, at instant t=1s, the sample from current data may be c = [3.03, 3.41, 3.27, 3.21, 3.04, 3.01, 3.01, 3.06, 2.9, 3.07]. The array will go through the LSTM autoencoder model which will produce a reconstructed array as illustrated by Fig. 9. The same MSE calculations are done for samples for the x-axis, y-axis, and z-axis acceleration. An example in Fig. 10 displays the outcome of MSE calculations for all four variables. It must be noted that this is only an example. As data from the motor is collected under different speed conditions, separate MSE calculations are done for the dataset from each speed level.

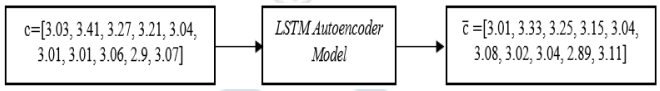


Fig. 9. A data sample being fed to the anomaly detection model.

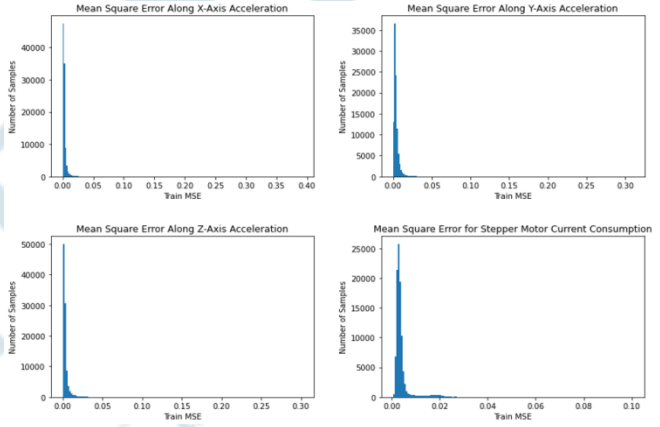


Fig. 10. A representation of MSE calculations on the training data.

We have already shown how MSE calculations are done, but how is the threshold determined? A threshold needs to be selected for the x-axis, y-axis, and z-axis acceleration and current consumption. The logical choice would be to select the maximum possible MSE value for each variable. However, at times the training data may contain a certain level of noise, which may cause a handful of samples to have a significantly high maximum value compared to the rest. Selecting these maximum values may not be the right choice for thresholds as even anomalous data may not display such high MSE. Therefore, we select thresholds by using a normal distribution curve.

In a normal distribution curve, a value which is three standard deviations (σ) above the mean (μ) of a dataset is considered to represent 99.7% of the samples as shown in Fig. 11. Therefore, three σ above the μ can be the threshold MSE value for each variable. Nevertheless, we aim for more inclusive thresholds and choose four σ above the μ which represent 99.9% of a data sample. As an example, thresholds for each variable when the stepper motor is rotating at 200 RPM are shown in Table 1. Different variable thresholds are obtained as the model is deployed at different motor speed levels.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

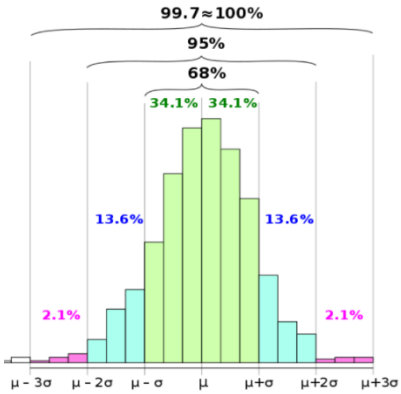


Fig. 11. A theoretical normal distribution curve.

Table 1. An example of MSE thresholds for each variable when the motor is rotating at 200 RPM.

X-axis	0.023458609192501197
Y-axis	0.026473839585112145
Z-axis	0.023598881032951696
Current Consumption	6.594044765752045e-05

Overall, for each of the four variables, if the MSE value of a sample is above the threshold, it is classified as an anomaly. All the calculations shown in this subsection are done autonomously by the LSTM autoencoder-based anomaly detection system. Once the system has completed training the model, it deploys the model on the training data to obtain these thresholds. These thresholds are used by the system to detect anomalies. It should be noted that the main system does not only detect the number of anomalies, but it also calculates the rate of anomalies that are occurring per minute. Basically, the system code samples the total number of anomalies that are occurring in a minute. So, if the total anomalies in a minute rises above a certain level (1 anomaly per minute in this case), the system notifies the user that the condition of the motor is degrading. The software part of the system which does all these necessary things autonomously is explained in the next sub-section.

F. PC User Interface

The PC user interface (UI) which gives users access to the unsupervised anomaly detection system is shown in Fig. 12. As shown in the image, the UI seems simple. It, however, has a wide array of functions. It is turned on after all the sensors have been connected to the motor and the Arduino board is plugged into USB port of the computer where the UI would be running. After it has been turned on, the user can initiate autonomous data collection by entering the duration of data collection in the input box labeled "Time", entering the speed of the motor in the "Speed" input box and press the "Load Data" button. The data is saved in an excel file which named according to the speed input: *[speed value]_speed.csv*.

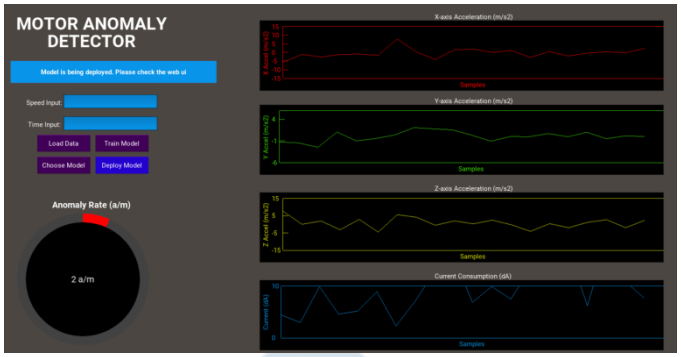


Fig. 12. The status of the PC UI when models are deployed.

Once the data collection is complete, the user has to click the "Train" button to begin training the LSTM autoencoder model. The system will automatically use the collected data for the specified speed level to initiate the training phase. When the model is model is complete, it is saved a *.hdf* file which is named according to the speed input to the UI. At this stage, the system automatically uses the trained model to find the MSE thresholds of each variable. This is done by deploying the model on the training dataset and carrying out the calculations and steps illustrated in the previous sub-section. The MSE threshold readings obtained are then saved in *.txt* file (also named according to the speed input).

Now the next stage would be to deploy the model. The user has to click the "Deploy" button to and select the desired model from the database as shown in Fig. 13. Once that is done, the specified model will be deployed to detect abnormalities in real-time. The status of the UI when models are deployed is shown in Fig. 12. The trend in data and the rate of anomalies can be observed by opening the system's web UI (which is a separate program file from the PC UI), which is illustrated in the next subsection.

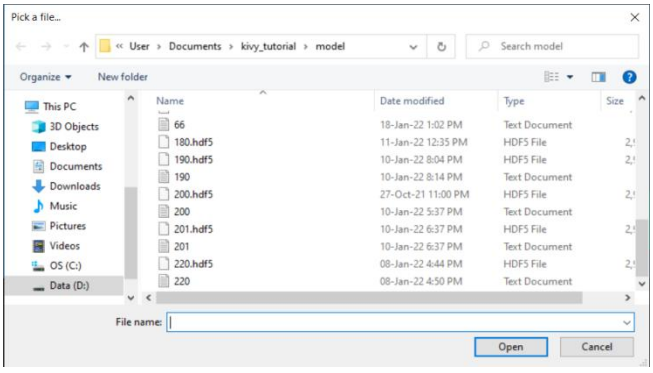


Fig. 13. The directory where the models are stored.

G. The Web User Interface

The web UI is made using HTML, CSS, and JavaScript code along with the Flask Python framework. MQTT protocols are also employed to get the remote data trends. These protocols are used to send the arrays for vibration and current consumption data from PC UI code and these arrays

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

are received by the web UI program, which displays them as graphs.

The web UI is primarily made so that the user can check the status of the electric motor from a remote location. It displays the trend in vibration and current consumption data along with the rate of anomalies occurring in the stepper motor. Next to the anomaly rate meter, as shown in Fig. 14, is the block that displays the current state of the motor. If the anomaly rate is 0 anomalies per minutes (a/m) or 1 a/m (in case of noise from the surroundings that can affect the vibration), the status block will display "Status Ok". In the case where the anomaly rate is between 2 a/m and 4 a/m, the status changes to "Abnormalities". If the anomaly rate goes beyond 4 a/m, the status changes to "Change Motor", the state of the motor would be severe. Appearances of the web UI are shown in Fig. 14.

Fig. 14. The anomaly detection system's web UI.

The web UI can be accessed from anywhere through a link. This UI is the one responsible for sending notifications to users. When the rate of abnormalities rises, the system sends an email and an SMS to the user, asking them to check the motor's condition via the web UI. The messages come with an accessible link that users can use to observe the motor's conditions. This web UI is responsive enough to be viewed on a phone screen. The anomaly detection system is tested on the hardware setup under varying conditions. The results of these experiments are explained in the next section.

H. Experiments under varying speed and load conditions

The LSTM autoencoder-based anomaly detection system is tested on experimental hardware setup under different load and speed conditions. The stepper motor is rotated at speed levels of 190 RPM (rotations per minute), 200 RPM, 210 RPM, and 220 RPM. At each level weights of 2.5kg, 3.75kg, 5kg, and 6.25kg are added to the hardware setup incrementally. These weights are placed in the basket tied to the slider as shown in Figure 5 to create the unbalanced conditions necessary for a bearing defect. As more weights are placed in the basket, there would be more imbalance, reminiscent of a worsening motor bearing defect. Therefore, as the weights increase the rate of anomalies would also rise. The purpose of these experiments is to show that the intelligent anomaly detection system presented in this paper does indeed work and can detect abnormalities in the motor when they occur.

In the experiments, each speed level has a different LSTM autoencoder model. It should be noted that the data and the model training are done using the PC UI as shown in the *Methodology* section. At first, the experimental setup is instructed to run at a particular speed, which it maintains through a PID controller. The PC UI is then initiated. After the specific duration (1 hour for all the experiments) and speed of the motor are entered into the input boxes, the "Load Data" button is clicked to begin automatic data collection. If the duration of this phase is over, an LSTM autoencoder model is trained using the specifically collected data by clicking the "Train Model" button. The training specifications are as mentioned in the last paragraph of *Section D* and were hardcoded into the main system code before the experiments. After the training is complete and the model is saved as explained in *Section F*, the specific model is selected from the database before clicking the deploy button. After the model is deployed, the system's web UI program can be run to observe the condition of the stepper motor. The anomaly rates as the weights are added are recorded. The outcome of the experiments is shown in the *Results* section.

III. RESULTS

As mentioned in the previously, experiments were carried out under different speed and load conditions. The main system was deployed on the hardware setup using the PC UI and the results of each experiment were observed through the web UI. Anomaly rates for each experiment are recorded in Table 2. The experimental results illustrated below are divided based on the speed levels and the load conditions.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 9, Issue 6, June 2022

Table 2: Results with varying load and speed conditions.

Speed (RPM)	Load (kg)	Anomaly Rate (a/m)	Status on Web UI
190	2.5	3	Abnormality
190	3.75	4	Change Motor
190	5	5	Change Motor
190	6.25	6	Change Motor
200	2.5	2	Abnormality
200	3.75	5	Change Motor
200	5	6	Change Motor
200	6.25	7	Change Motor
210	2.5	2	Abnormality
210	3.75	3	Abnormality
210	5	4	Change Motor
210	6.25	5	Change Motor
220	2.5	3	Abnormality
220	3.75	5	Change Motor
220	5	6	Change Motor
220	6.25	7	Change Motor

As shown by the above images, the rate of anomalies per minute increases as the weight acting on the stepper motor increases. This is proof that the system is capable of detecting anomalies. According to the above images, the system can also indicate worsening defects (bearing defects in this case). As the anomaly rates rise, the web UI sends an SMS notification to the user. Thus, this LSTM autoencoder-based anomaly detection system can be used for fault diagnosis of motors. Overall, based on these results it is clear that this system meets the objectives. It can analyze vibration and current consumption data and detect abnormalities when there are underlying issues. Through the autonomous data collection, model training, and model deployment features in the PC UI it can adapt to different speed and load conditions. The system managed to train specific LSTM autoencoder models for each speed level to detect anomalies. Last but not least, the web UI of the system works well enough to help users keep track of the motor's conditions in real-time. Thus, we can say that the LSTM autoencoder-based anomaly detection system has worked successfully.

IV. CONCLUSION

Having an effective PdM system is essential to reduce unexpected machine downtimes, maintenance costs, and calamitous failures. Unlike conventional maintenance strategies, PdMs are less prone to errors and are less exhaustive. Furthermore, they have the potential to be autonomous. The FDM systems are a branch of PdM. They have the potential to detect underlying issues in electric motors, which are crucial components in many modern-day machines before they break down and cause financial losses. Unfortunately, the majority of FDM systems proposed so far are based on supervised learning methods. Therefore, to develop an FDM system an individual would have to collect and annotate data for hours. This can require manpower and time which a lot of companies are unable to allocate. Besides this, most of the proposed systems were developed to classify a single type of motor fault, whereas multiple motor issues may occur. Simulating the many fault conditions to train supervised models may be challenging. On top of this, different models would have to be trained depending on the load and speed conditions of a motor. Overall, there are an array of challenges when it comes to implementing FDM systems. However, what most people forget is that industries only wish to know if their motors are working properly and have no need for specific fault classification systems as proposed by previous works. Detecting abnormalities is one way to detect underlying motor issues. Unlike supervised methods, they can be unsupervised. Thus, anomaly detection systems would not require manual annotation of data, which is why they will consume less time and manpower. Hence, to overcome the issues of most proposed FDM systems, this paper presents an LSTM autoencoder-based anomaly detection system. The presented system detects anomalies by analyzing motor vibration and current consumption data. It is unsupervised and comes with a PC UI that can allow users to autonomously collect data, and train and deploy models. The system also comes with a web UI which helps users to observe the trend in data along with the rate of anomalies and the status of the motor.

To test this anomaly detection system, an experimental hardware setup, which is a slider controlled by a stepper motor, is developed. The slider has an integrated PID controller that helps the setup to maintain a given speed. An accelerometer and a current sensor are attached to the stepper motor for data collection. The purpose of this slider is to simulate a motor bearing defect through the addition of unbalanced weights. As illustrated in Section 2.2, as the weights are added to the slider, there are variations in data, especially in current consumption. This is indicative that the experimental setup is suitable for testing the anomaly detection system.

The system makes use of an LSTM autoencoder model to analyze vibration and current consumption data patterns from the stepper motor. It is chosen as it can reconstruct time-series signals. By finding MSE errors between the model's original input and reconstructed output signals, it is possible to find anomalies. There is an MSE threshold for each of the four data variables: x-axis, y-axis, and z-axis acceleration, and current consumption. The system calculates these thresholds autonomously using an LSTM autoencoder model, as explained in the *Methodology* section. Through the

application of the thresholds, the anomaly rates are calculated.

The overall anomaly detection system along with all of its features is deployed on the slider's stepper motor at different speed and load conditions. Outcomes of these experiments are illustrated in the *Results* section. As shown by the images of the web UI, the anomaly rates for each speed level rise as more unbalanced weights are added to the slider. As the anomaly rates increase the system sends an SMS message to the user along with a link to the web UI, that allows the user to observe the motor's conditions.

Overall, based on the results it is evident that the system meets all the requirements to be an effective FDM system. It can detect anomalies autonomously by analyzing vibration and current consumption data. Its features allow it to adapt to different motor conditions. Finally, the system's web IU can send notifications and help users to keep track of the motor's conditions. The presented system works as expected and unlike many of the presented systems in literature, it does not require users to manually collect and annotate data for fault classification. Its autonomous data collection, model training, and model deployment features can make the system viable for industries.

While the presented system has proven to be successful, it needs to be studied in more detail. For this paper, experiments are only carried out on a stepper motor. In future works, different motors may be used to evaluate the effectiveness of this anomaly detection system. Perhaps next time the system could be tested on a brushless DC motor. Should future experiments prove to be just as successful, there is no doubt that the LSTM autoencoder-based anomaly detection system can be implemented in companies for fault diagnosis of electric motors. If this is possible, it will not only be a strong contribution to the manufacturing industries, where motors are frequently used, but also to the automobile and aviation industries.

REFERENCES

[1]. J. Wan et al., "A Manufacturing Big Data Solution for Active Preventive Maintenance," IEEE Transactions on Industrial Informatics, vol. 13, no. 4, pp. 2039–2047, 2017, doi: 10.1109/TII.2017.2670505.

[2]. G. A. Susto, A. Beghi, and C. De Luca, "A Predictive Maintenance System for Epitaxy Processes Based on Filtering and Prediction Techniques," IEEE Transactions on Semiconductor Manufacturing, vol. 25, no. 4, pp. 638–649, 2012, doi: 10.1109/TSM.2012.2209131.

[3]. A. Jezzini, M. Ayache, L. Elkhansa, B. Makki, and M. Zein, "Effects of predictive maintenance (PdM), Proactive maintenance (PoM) amp; Preventive maintenance(PM) on minimizing the faults in medical instruments," in 2013 2nd International Conference on Advances in Biomedical Engineering, 2013, pp. 53–56, doi: 10.1109/ICABME.2013.6648845.

[4]. V. Kavana and M. Neethi, "Fault Analysis and Predictive Maintenance of Induction Motor Using Machine Learning," in 2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), 2018, pp. 963–966, doi: 10.1109/ICEECCOT43722.2018.9001543.

[5]. S. Altaf, M. W. Soomro, and M. S. Mehmood, "Fault Diagnosis and Detection in Industrial Motor Network Environment Using Knowledge-Level Modelling Technique," Modelling and Simulation in Engineering, vol. 2017, p. 1292190, 2017, doi: 10.1155/2017/1292190.

[6]. K. Hendrickx et al., "A general anomaly detection framework for fleet-based condition monitoring of machines," Mechanical Systems and Signal Processing, vol. 139, p. 106585, 2020, doi: https://doi.org/10.1016/j.ymssp.2019.106585.

[7]. B. Song and H. Shi, "Fault Detection and Classification Using Quality-Supervised Double-Layer Method," IEEE Transactions on Industrial Electronics, vol. 65, no. 10, pp. 8163–8172, 2018, doi: 10.1109/TIE.2018.2801804.

[8]. Y. Lei, F. Jia, J. Lin, S. Xing, and S. X. Ding, "An Intelligent Fault Diagnosis Method Using Unsupervised Feature Learning Towards Mechanical Big Data," IEEE Transactions on Industrial Electronics, vol. 63, no. 5, pp. 3137–3147, 2016, doi: 10.1109/TIE.2016.2519325.

[9]. C. Sobie, C. Freitas, and M. Nicolai, "Simulation-driven machine learning: Bearing fault classification," Mechanical Systems and Signal Processing, vol. 99, pp. 403–419, 2018, doi: https://doi.org/10.1016/j.ymssp.2017.06.025.

[10]. K. C. D. Kompella, V. G. R. Mannam, and S. R. Rayapudi, "DWT based bearing fault detection in induction motor using noise cancellation," Journal of Electrical Systems and Information Technology, vol. 3, no. 3, pp. 411–427, 2016, doi: https://doi.org/10.1016/j.jesit.2016.07.002.

[11]. K. Hendrickx et al., "A general anomaly detection framework for fleet-based condition monitoring of machines," Mechanical Systems and Signal Processing, vol. 139, p. 106585, 2020, doi: https://doi.org/10.1016/j.ymssp.2019.106585.

[12]. V. Vercruyssen, W. Meert, G. Verbruggen, K. Maes, R. Bäumer, and J. Davis, "Semi-Supervised Anomaly Detection with an Application to Water Analytics," in 2018 IEEE International Conference on Data Mining (ICDM), 2018, pp. 527–536, doi: 10.1109/ICDM.2018.00068.

[13]. S. A. Chopade, J. A. Gaikwad, and J. V. Kulkarni, "Bearing fault detection using PCA and Wavelet based envelope analysis," Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, pp. 248–253, 2017, doi: 10.1109/ICATCCT.2016.7912002.

[14]. I. H. Ozcan, O. C. Devecioglu, T. Ince, L. Eren, and M. Askar, "Enhanced bearing fault detection using multichannel, multilevel 1D CNN classifier," Electrical Engineering, 2021, doi: 10.1007/s00202-021-01309-2.

[15]. S. Shao, R. Yan, Y. Lu, P. Wang, and R. X. Gao, "DCNN-Based multi-signal induction motor fault diagnosis," IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 6, pp. 2658–2669, 2020, doi: 10.1109/TIM.2019.2925247.

[16]. R. Wang, Z. Feng, S. Huang, X. Fang, and J. Wang, "Research on voltage waveform fault detection of miniature vibration motor based on improved WP-LSTM," Micromachines, vol. 11, no. 8, 2020, doi: 10.3390/MI11080753.

[17]. Arun, V., Prabaharan, N.(2019). "Induction motor drive with trinary dc source asymmetrical inverter", International

Journal of Recent Technology and Engineering, 2019, Vol. 8, No.2, pp. 5484–5490. DOI: 10.35940/ijrte.B2527.078219.

[18]. K. Bajaj, D. K. Singh, and M. A. Ansari, "Autoencoders Based Deep Learner for Image Denoising," Procedia Computer Science, vol. 171, pp. 1535–1541, 2020, doi: https://doi.org/10.1016/j.procs.2020.04.164.

[19]. A. Pol et al., "Anomaly detection using Deep Autoencoders for the assessment of the quality of the data acquired by the CMS experiment," EPJ Web of Conferences, vol. 214, p. 6008, Jan. 2019, doi: 10.1051/epjconf/201921406008.

[20]. E. Principi, D. Rossetti, S. Squartini, and F. Piazza, "Unsupervised electric motor fault detection by using deep autoencoders," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 2, pp. 441–451, 2019, doi: 10.1109/JAS.2019.1911393.

[21]. Y. Huang, C. H. Chen, and C. J. Huang, "Motor fault detection and feature extraction using rnn-based variational autoencoder," IEEE Access, vol. 7, pp. 139086–139096, 2019, doi: 10.1109/ACCESS.2019.2940769.

[22]. G. Dong, G. Liao, H. Liu, and G. Kuang, "A Review of the Autoencoder and Its Variants: A Comparative Perspective from Target Recognition in Synthetic-Aperture Radar Images," IEEE Geoscience and Remote Sensing Magazine, vol. 6, pp. 44–68, Sep. 2018, doi: 10.1109/MGRS.2018.2853555.