IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 2, 2020, accepted June 26, 2020, date of publication July 2, 2020, date of current version July 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006788

An Effective Predictive Maintenance Framework for Conveyor Motors Using Dual Time-Series Imaging and Convolutional Neural Network in an Industry 4.0 Environment

KAHIOMBA SONIA KIANGALA AND ZENGHUI WANG, (Member, IEEE)
Department of Electrical and Mining Engineering, University of South Africa, Johannesburg 1710, South Africa

Corresponding author: Zenghui Wang (wangzengh@gmail.com)

This work was supported in part by the South African National Research Foundation under Grant 112108 and Grant 112142, and in part by the Tertiary Education Support Program (TESP) of South African ESKOM.

ABSTRACT The ascent of Industry 4.0 and smart manufacturing has emphasized the use of intelligent manufacturing techniques, tools, and methods such as predictive maintenance. The predictive maintenance function facilitates the early detection of faults and errors in machinery before they reach critical stages. This study suggests the design of an experimental predictive maintenance framework, for conveyor motors, that efficiently detects a conveyor system's impairments and considerably reduces the risk of incorrect faults diagnosis in the plant; We achieve this remarkable task by developing a machine learning model that classifies whether the abnormalities observed are production-threatening or not. We build a classification model using a combination of time-series imaging and convolutional neural network (CNN) for better accuracy. In this research, time-series represent different observations recorded from the machine over time. Our framework is designed to accommodate both univariate and multivariate time-series as inputs of the model, offering more flexibility to prepare for an Industry 4.0 environment. Because multivariate time-series are challenging to manipulate and visualize, we apply a feature extraction approach called principal component analysis (PCA) to reduce their dimensions to a maximum of two channels. The time-series are encoded into images via the Gramian Angular Field (GAF) method and used as inputs to a CNN model. We added a parameterized rectifier linear unit (PReLU) activation function option to the CNN model to improve the performance of more extensive networks. All the features listed added together contribute to the creation of a robust future proof predictive maintenance framework. The experimental results achieved in this study show the advantages of our predictive maintenance framework over traditional classification approaches.

INDEX TERMS Convolutional neural network (CNN), Gramian angular field (GAF), industry 4.0 (I40), predictive maintenance, principal component analysis (PCA), smart manufacturing, time-series imaging.

I. INTRODUCTION

The recent explosion of smart manufacturing applications, the Internet of things (IoT), and big data has considerably increased the amount of data collected and analyzed in different areas such as health care, transportation, power energy, food and beverage, multimedia, environment, finance, and logistics. Several types of predictions, production forecasting, fault detection and, predictive maintenance result from analyzing various datasets [1], [2]. One of the most common data types collected in this new growing era of Industry 4.0 is time-series data. Time-series data are known as observations sequentially recorded over time [3], [4].

Time-series data are intensively analyzed, as a preventive tool, in the manufacturing industry where unforeseen failures of machinery can conduct to very long production downtime and losses. Studying and analyzing data to detect faults and threats in devices before they occur and taking appropriate measures to reduce the risk of failures is called ''predictive maintenance'' [5]. As per [6], predictive maintenance is an ensemble of activities that detect any abnormal physical condition changes in equipment (signs of failure) to carry out the

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

required maintenance tasks to boost the service life of equipment without increasing the risk of failure. For the past years, predictive maintenance has been subject to much research to bring improvement. One of the current innovative trends for this concept is the use of machine learning (ML) techniques in combination with advanced technological concepts to offer better predictive maintenance results.

Machine learning (ML) is a field of Artificial Intelligence (AI) to extricate useful insights from various data (time-series data) [7] through some of the following paradigms: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning [8]. It is also commonly known as a study that offers machines different means and ways to make correct decisions on their own and execute tasks without explicit assistance from human beings. Deep Learning is a branch of ML that has the capability of extracting data representation. Some popular deep learning methods are Artificial Neural Network (ANN), Convolution Neural Networks (CNN), Deep Belief Network, Recurrent Neural Networks, and Stacked Auto-Encoders [9].

In this research, we focus on CNN, which is a deep learning technique that tries to imitate the operations of a human brain, especially its ability to recognize and classify objects based on their appearances. This feature has made CNN the conventional method used for image classification and identification [10]. In 2015, [11] initiated an inventive approach that improved classification and imputation by encoding univariate time-series (UTS) data to images and using them as inputs to CNN models. The concept of computer vision introduced the transformation of time-series into images. By learning spatially invariant features from raw time series (inputs to the model), the CNN method can reduce the risks of losing temporal information and those that the features learned are no longer time-invariant, which are with the traditional multilayer-perceptron approach [12]. The outcome of this study generated better results than traditional machine learning techniques for classification, such as decision tree (DT), random forest (RF), or Support Vector Machine (SVM). Since then, fewer more studies were conducted in the same vision utilizing the basis of time-series imaging encoding and deep learning approaches to ameliorate classification modeling in various sectors. Reference [13] developed a similar framework that uses Relative Position Matrix with CNN. The method was named RPMCNN and was used to perform the classification by transforming 2D images from time-series data received as inputs. Their results displayed improved performances. In the manufacturing sector, an approach was introduced by [14] using multivariate time-series (MTS) data as input to a classification of Tool wear for a CNN model. Because of the large volume of MTS data and in order to ease data processing, this approach divided MTS inputs into three channels before being converted to images and fed into the CNN model. Reference [3] conducted another research in that direction by converting MTS data to colored images and feeding them as inputs of a CNN model for sensor classification. Their research encodes MTS data into multiple images combined into a single bigger image used as an input to a CNN model.

Our research takes a step further on previous work done in the manufacturing sector on this innovative concept by developing an experimental framework that:

1) Generates accurate predictive maintenance flags for conveyor motors by classifying whether observed system parameters inputs are threats or not.

2) Combines the use of UTS and MTS in one single platform to increase the flexibility of the system. No need to have separate models.

3) Facilitates inputs and manipulations of MTS data in CNN by reducing their size to two channels through a feature extraction method called principal component analysis (PCA).

4) Offers an option for a future proof CNN model by using parameterized rectifier linear unit configuration (PReLU) to improve the performance of larger networks.

Our paper is structured as follows: Section 2 presents a literature review of some concepts such as time-series, deep learning, predictive maintenance with machine learning, Imaging time-series for classification. Section 3 describes the methodologies, technological approaches, and architecture of our predictive maintenance framework. Section 4 presents the experimental results obtained, and the conclusion and suggestions for future research are provided in Section 5.

II. LITERATURE REVIEW

A. TIME-SERIES DATA

As mentioned previously, time-series can be defined as a sequence of observations recorded over successive time points [15]. Time series data can be grouped into two main categories: Univariate Time Series (UTS) and Multivariate Time Series (MTS). UTS are time series composed of a single variable observed over a regular period of time. MTS are those made of two or more variables recorded over a successive period of time [16].

Equation (1) is a mathematical representation of UTS defined as follows:

$$B = [b_1, b_2, b_3, \cdots, b_n, \cdots b_t] \tag{1}$$

where $b_n \in \mathbb{R}$, $t \in \mathbb{N}$ and represents the size of the time series data.

On the other hand, (2) is an expression for MTS.

$$D = [B_1, B_2, B_3, \cdots, B_i, \cdots B_m] \tag{2}$$

where $m \in \mathbb{N}$ and represents the size of the MTS, m is also equal to the number of univariate time series in D, i is the unique position identification for each UTS in D. As per (2), D contains several UTS similar to those defined in (1). For a MTS, D, regrouping a number of UTS, B, a single UTS object can be defined by (3) as:

$$B_i = \left[b_{i(1)}, b_{i(2)}, b_{i(3)}, \cdots, b_{i(n)}, \cdots b_{i(t)}\right] \tag{3}$$

where $t \in \mathbb{N}$ and is the size of the UTS [17], [18], i is the unique position identification for each UTS in the MTS.

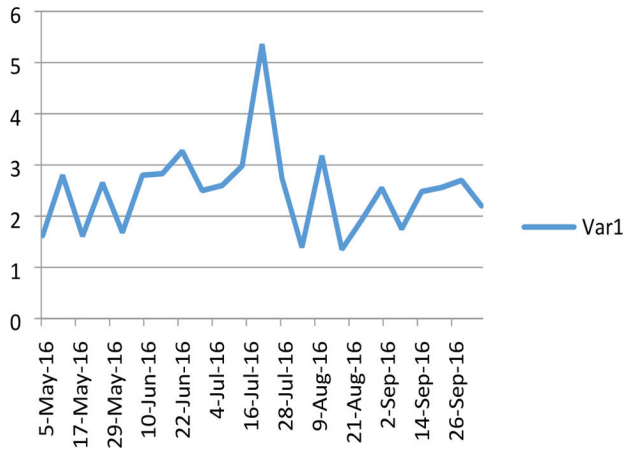

FIGURE 1. Example of a UTS graphical representation.

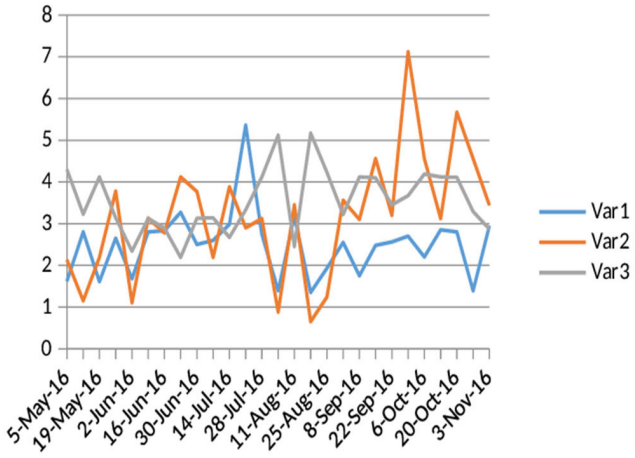

FIGURE 2. Example of a MTS graphical representation.

From these mathematical representations, we can conclude that UTS represents a vector while MTS represents a matrix (a combination of multiple vectors). Based on the above parameters, let assume a MTS D where t = 4 and m = 4. The MTS data can be defined by (4) in a matrix format as:

$$D = \begin{pmatrix} b_{11} & b_{21} & b_{31} & b_{41} \\ b_{12} & b_{22} & b_{32} & b_{42} \\ b_{13} & b_{23} & b_{33} & b_{43} \\ b_{14} & b_{24} & b_{34} & b_{44} \end{pmatrix} \tag{4}$$

The graphical representation of UTS and MTS are displayed in Fig. 1 and Fig. 2 respectively:

In Fig.1, a single time series variable (UTS) value, ranging from 0 to 6, named 'Var1', is recorded between the 05 May 2016 to the 26 September 2016. Var1 represents any other parameter measured over an interval. The main difference between Fig.1 and Fig.2 is simply the number of variables monitored over time. Fig.2 recorded three different variables: Var1, Var2, and Var3 (MTS). Var1 to Var3 represents any parameter values observed over the same period. More complex MTS has more than three variables.

B. CONVERTING TIME-SERIES DATA TO IMAGES

Transforming time series to images is one type of data transformation. An exciting data transformation approach is to reduce the size or dimension (dimension reduction) of massive volumes datasets from high dimensional data of more than three features to only 2 (2-D) or sometimes 3-dimensional (3-D) providing a better understanding of the data, especially when it comes to visualization [24], [25]. Another data transformation method that seems to be opposite to the previous one is "dimension augmentation" as it involves increasing the size of a particular dataset; for example, going from a 1-dimensional (1-D) data into2-D, or even 3-D. Dimension augmentation is a crucial step, especially when considering using a CNN model, which we intend to in this study. Reference [11] introduced an approach named Gramian Angular Field (GAF) for encoding time series into images to improve classification and imputation. GAF uses a polar coordinates-based matrix to encodes time series into image since they have the advantage of preserving temporal correlation, unlike the Cartesian coordinate [11]. GAF can generate two types of images: Gramian angular summation field (GASF) and Gramian angular differential field (GADF). The steps to obtain GAF images are as follows:

1) NORMALIZING TIME SERIES DATA INPUT

The original time series (1) (as described in the previous section) is scaled or normalized to values in the intervals of $[-1, 1]$. The normalization method is defined in (5).

$$\widetilde{b_{-1}}^{i} = \frac{(b_i - max(B)) + (b_i - min(B))}{max\,(B) - min(B)} \tag{5}$$

where $\widetilde{b_{-1}}^{i}$ is the scaled or the normalized value of each original time series observation b_i.

2) CONVERTING SCALED TIME-SERIES DATA TO POLAR COORDINATES

The second step of imaging time series with the GAF method consists of representing the normalized time series $\tilde{B}$ in polar coordinates. The polar coordinates are computed by finding the angular cosine of each normalized value and the time stamp which represented as a radius. The polar coordinates are defined by (6) and (7):

$$\theta = \arccos(\tilde{b_i}) \tag{6}$$

where $-1 \leq \tilde{b_i} \leq 1, \tilde{b_i} \in \tilde{B}$

$$r = \frac{\mathrm{t_i}}{\mathrm{N}}, \quad \mathrm{t_i} \in \mathbb{N} \tag{7}$$

In (6), θ represents the time series value of each observation in the polar coordinates format. In (7), t_i represents the time stamp of the time series data and N is a factor (a constant) that stabilizes the polar coordinate system's space.

3) FINDING THE GRAMIAN ANGULAR SUMMATION/DIFFERENCE FIELD (GASF & GADF)

After obtaining the polar coordinates of the time series, we make use of trigonometric sum and difference to find

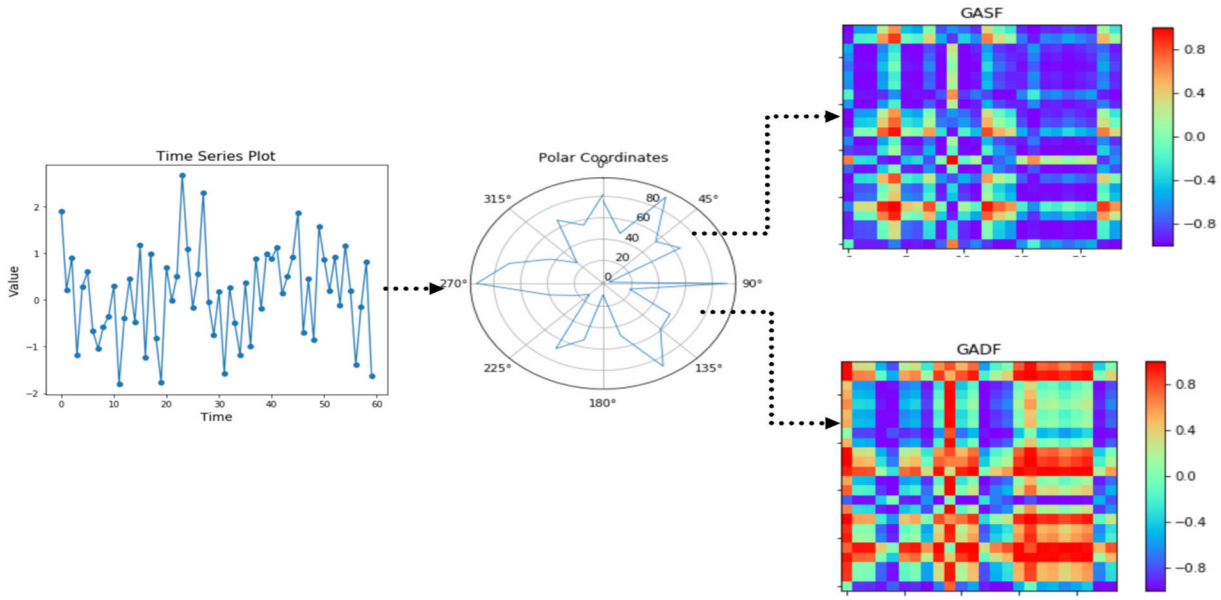

FIGURE 3. From time series to Gramian angular fields process.

spatial correlation between each polar point and determine their GASF and GADF. GASF mathematical representation is presented in (8) and (9). GADF is defined in (10) and (11).

$$GASF = \left[\cos\left(\theta_i + \theta_j\right)\right] \tag{8}$$

$$GASF = \tilde{b}' \cdot \tilde{b} - \sqrt{1 - \tilde{b}^{2'}} \cdot \sqrt{1 - \tilde{b}^2} \tag{9}$$

$$GADF = \left[\sin\left(\theta_i - \theta_j\right)\right] \tag{10}$$

$$GADF = \sqrt{1 - \tilde{b}^{2'}} \cdot \tilde{b} - \tilde{b}' \sqrt{1 - \tilde{b}^2} \tag{11}$$

A popular mathematical representation of GAF is done in a matrix format and defined by (12) and (13):

$$GASF = \begin{pmatrix} \cos(\theta_1 + \theta_1) & \cdots & \cos(\theta_1 + \theta_n) \\ \vdots & \ddots & \vdots \\ \cos(\theta_m + \theta_1) & \cdots & \cos(\theta_m + \theta_n) \end{pmatrix} \tag{12}$$

$$GADF = \begin{pmatrix} \sin(\theta_1 - \theta_1) & \cdots & \sin(\theta_1 - \theta_n) \\ \vdots & \ddots & \vdots \\ \sin(\theta_m - \theta_1) & \cdots & \sin(\theta_m - \theta_n) \end{pmatrix} \tag{13}$$

A graphical representation of steps to convert time series to GAF is displayed in Fig.3.

In this research, we focus on the GAF method for image encoding since it preserves the temporal correlation of time series data inputs which is needed for our predictive maintenance framework.

C. CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional neural network (CNN) is a deep learning algorithm successfully used for image classification problems. The outstanding performance of CNN in image classification (computer vision) is due to its ability to extracting meaningful spatial correlation and create features information from input data used to detect patterns. The animals' visual cortex was the inspiration behind the CNN concept and introduced by Hubel and Wiesel, two neurophysiologists who did many types of research on visual cortical neurons of monkeys and cats [26]. The first modern CNN framework was called LeNet and was published by [27]. After this first model, several other successful architectures such as ResNet [28], AlexNet [10], VGGNet [29], Inception v3 etc. [30]

An underlying CNN architecture has the following layers:

1) A CONVOLUTIONAL LAYER

This layer extracts the input image features by using some filters (feature detectors) and generating a smaller size image containing the original input image features. The result of the convolutional layer is called a feature map. Before going to the next layer, in most CNN architectures, an activation function is applied to the feature maps to increase the non-linearity of the image (useful to avoid linearity in images since most images have non-linear features predominantly). One of the most popular activation functions used in deep learning for the past few years in the Rectifier Linear Unit (ReLU) [31].

2) A POOLING LAYER

The pooling layer's objectives are to generate a spatial invariant feature for the image (the ability to recognize the image in positions different than the input image) and reduce the size of the feature maps. One standard pooling method used is "Max Pooling" [32]. Many other convolutional and pooling

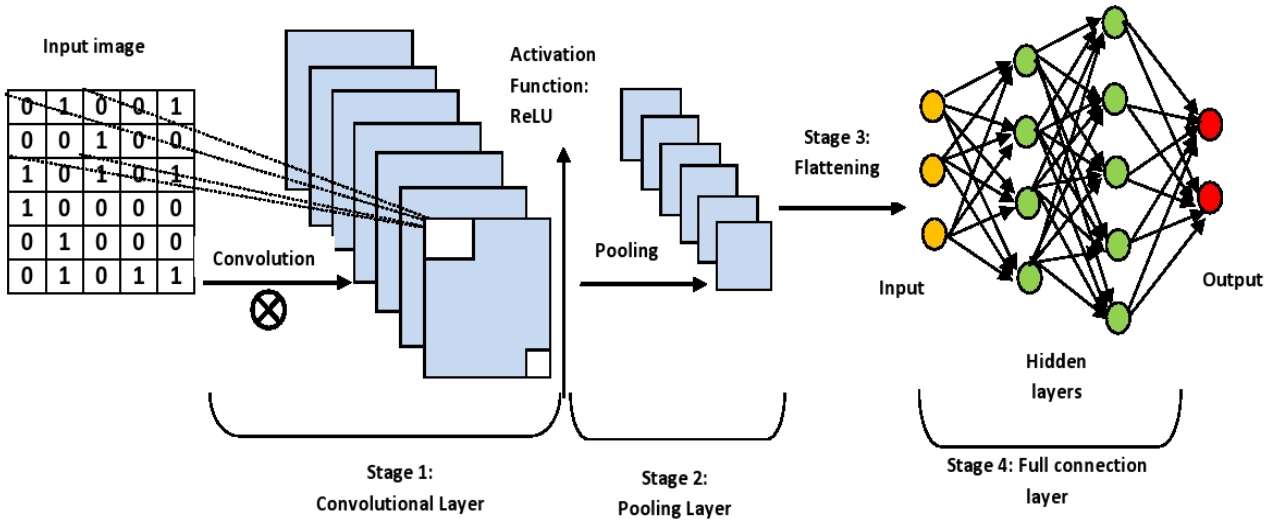

FIGURE 4. Basic structure of CNN architecture.

layers can be added before the flattening layer to improve the accuracy of a CNN model.

3) A FLATTENING LAYER

The flattening layer converts the pooled feature map matrix (2D) into a vector(1D) input for the neural network (next layer) [3].

4) A FULL CONNECTION LAYER

The full connection layer is a neural network composed of several neurons' layers interconnected through the synopsis and converging to the final outputs. The full connection layer is where all the classification intelligence of the CNN happens. The first neurons layer receives its input from the previous flattening layer and goes through several hidden layers before producing the results.

Note: One way to improve a CNN model accuracy is to pass through the model forward and backward several times by adjusting the weight of the inputs (iterating the dataset) based on obtained output results until we achieve the desired accuracy. The number of time the dataset is iterated is called Epochs [33].

A graphical representation of the basic structure of a CNN architecture is presented in Fig.4.

D. PREDICTIVE MAINTENANCE APPROACHES USING MACHINE LEARNING TECHNIQUES

Predictive maintenance is one advantageous approach to ensure smooth and reliable operations of production processes. For the past years, researchers conducted many studies to improve predictive maintenance techniques. One innovative trend introduced in the research field is the use of machine learning techniques to ameliorate this concept's outcomes. A study was proposed by [19] to apply ML for the predictive maintenance of gas turbines. They focused on using SVM and Regularized Least Square (RLS) to predict the appropriate maintenance time of the gas turbines based on its speed, compressor decay, and gas turbines decay. The results showed that the SVM outperformed the RLS model. Another method proposed by Leahy *et al.* [20] uses SVM to perform Predictive maintenance on Wind Turbines. Their approach used ML to create a classification model for six faults in wind turbines: fault/no-fault, feeding faults, air cooling faults, excitation faults, generator heating faults, and mains failure. By using SVM hyperparameter optimization by randomized search, they reached better results on detecting generator heating faults, classifying correctly 100% of the cases. However, on the fault/no-fault dataset, they got only 90%recall and 8% precision. Susto *et al.* [21] introduced another ensemble approach to detect the best moment for tungsten filaments replaced during ion implantation. It is a step in the process of manufacturing semiconductors. The authors tested SVMs ensembles Predictive Maintenance with K-Nearest Neighbor (KNN) and Predictive Maintenance with SVM; the predictive maintenance with SVM gave slightly better results than the KNN approach. In [22], the authors used the random forest (RF) ML technique to generate a predictive maintenance approach for a cutting machine. The RF model used different rotor status of the cutting machine to perform classification in the predictive maintenance scheme. Kulkarni *et al.* [23] worked on a refrigeration and cold storage system by developing an ML base approach that performs predictive maintenance by detecting early faults on the machinery involved in the refrigeration. They apply a feature extraction step in the pre-processing phase of the model, which consisted of learning the pattern of the dataset and seasonality decomposition by dynamic time wrapping and clustering. They also built an RF classifier to recognize if the pattern was abnormal or not.

Following the same path to improve the quality of predictive maintenance approaches implemented for systems, our study applies CNN modeling that can extract feature representation to offer better results than traditional ML techniques. Unlike traditional ML techniques such as RF or SVM. CNN has the advantage of accommodating a very high number of features by quickly determining which ones have higher weights (more influential to the system) than the others, therefore eliminating unnecessary ones. In this era of IoT and Big data where massive amounts of data are available every day, it is convenient to integrate such a feature into modeling techniques.

Having a sound theoretical background on all useful concepts used in this study, let us have a detailed look at the methodology applied to construct the predictive maintenance framework. This methodology focuses on reducing the dataset dimension (PCA) and using CNN to achieve accurate classification.

III. PREDICTIVE MAINTENANCE FRAMEWORK METHODOLOGY

This experimental predictive maintenance framework aims to classify conveyor motor states as dangerous or not dangerous by encoding time series as images and feeding them into a CNN model that performs the classification task. The framework consists of the following stages:

1) FEEDING STAGE

In order to accommodate dual time-series types, we design this stage is responsible for the separation of MTS and UTS. This stage has two inputs. The feeding stage has a sub-stage for MTS data inputs. The sub-stage is called **"Dimensionality Reduction Stage"** and aims to reduce the size of MTS inputs to two channels using an approach called principal component analysis (PCA). By reducing the size of MTS data, the system's complexity decreases, and the performance improves (data processing volume reduces considerably).

2) IMAGING STAGE

At this stage, time series received from the feeding step: either a UTS or a Reduced MTS are converted into images using the GAF method.

3) CNN CLASSIFICATION MODELING STAGE

This stage receives encoded images from the previous step and performs a classification task using the CNN method. In this research, we add an option in the CNN model that uses the Parameterized Rectifier Linear Unit (PReLU) activation function to improve the non-linearity feature of input images and to achieve better accuracy at the output when using extensive input networks. Since we built our predictive model for small manufacturing industries, the performance results obtained using both CNN with classic rectifier linear unit (ReLU) and PReLU are very much similar.

Note: Although the performance improvement between CNN models with ReLU and those using PReLU has been proven by some authors to be very small (about 1% to 2% accuracy improvement), this could make a massive difference in the manufacturing environment where availability of systems and machines is essential to production. A 1% more accuracy could be the information needed to avoid chaos in the plant.

A. PRINCIPAL COMPONENT ANALYSIS (PCA) TECHNIQUE

PCA is an unsupervised learning method, it means that we don't make use of the dependent variable to perform its operations. A. Yunusa-Kaltungo *et al.* [54] describe a similar approach to reduce data dimension for fault diagnostic on rotating machines. To achieve dimensionality reduction using PCA, we go through the following steps:

1) PRE-REDUCTION OF DATASET DIMENSION

In this study, we consider datasets of time series variables, which are observations of different conveyor motor parameters indicating threats to the system. These could be observations of any other system. Our experimental data is composed of 12 parameters, 11 observations: Vibration speed, Motor torque, Acceleration, Motor Speed, Air pressure, Product Weight, Deceleration, Current, Belt tension, Motor tension, Temperature and one outcome which is the type of Fault detected in the system. Each parameter has about 15,000 observations or values recorded during a specific interval. We express the overall dataset as the expression $p + 1$, with p being the number of observations or independent time series variables and one the number of the dependent variable or the label (In our case, the type of faults generated). We discard the number of label one and remain with p as the new dimension of our dataset, in this case, p=11.

2) CALCULATE THE AVERAGE OF EVERY DIMENSION OF THE NEW DATASET

Since the new size of our dataset is $p = 11$, the dataset is composed of eleven time series variables or eleven vectors of observations. In this research, they can be detailed as follows:

$$\begin{aligned}
\text{P1(vibration speed)} &= [p_{11}, p_{12}, \cdots, p_{1n}] \\
\text{P2(motor torque)} &= [p_{21}, p_{22}, \cdots, p_{2n}] \\
\text{P3(acceleration)} &= [p_{31}, p_{32}, \cdots, p_{3n}] \\
\text{P4(motor speed)} &= [p_{41}, p_{42}, \cdots, p_{4n}] \\
\text{P5(air pressure)} &= [p_{51}, p_{52}, \cdots, p_{5n}] \\
\text{P6(product weight)} &= [p_{61}, p_{62}, \cdots, p_{6n}] \\
\text{P7(deceleration)} &= [p_{71}, p_{72}, \cdots, p_{7n}] \\
\text{P8(current)} &= [p_{81}, p_{82}, \cdots, p_{8n}] \\
\text{P9(belt tension)} &= [p_{91}, p_{92}, \cdots, p_{9n}] \\
\text{P10(motor tension)} &= [p_{101}, p_{102}, \cdots, p_{10n}] \\
\text{P11(temperature)} &= [p_{111}, p_{112}, \cdots, p_{11n}]
\end{aligned}$$

where *n* is the length of each time series variable.

In this experiment let us assume $n = 15{,}000$, we can generate a matrix (14) of size $p \times n$,$(11 \times 15{,}000)$, representing

TABLE 1. Variance-covariance vector relationship.

	P1	P2	...	P11
P1	VC(P1,P1)	VC(P1,P2)	...	VC(P1,P11)
P2	VC(P2,P1)	VC(P2,P2)	...	VC(P2,P11)
⋮	...	...	...	...
P11	VC(P11,P1)	VC(P11,P2)	...	VC(P11,P11)

the new dataset as:

$$D = \begin{pmatrix} p_{11} & \cdots & p_{61} & \cdots & p_{111} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{17500} & \cdots & p_{67500} & \cdots & p_{117500} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{115000} & \cdots & p_{615000} & \cdots & p_{1115000} \end{pmatrix} \quad (14)$$

The average of each dimension of the dataset D can be computed by equation (15):

$$AvgP1 = \frac{\sum_{i=1}^{n} (\mathrm{P_{1n}})}{\mathrm{n}} = \overline{\mathrm{P1}} \quad (15)$$

From (14) and (15), the average matrix originating from D can be summarized as follows:

$$\bar{D} = [\frac{\sum_{i=1}^{n} (p_{1n})}{n} \cdots \frac{\sum_{i=1}^{n} (p_{6n})}{n} \cdots \frac{\sum_{i=1}^{n} (p_{11n})}{n}] \quad (16)$$

$$\bar{D} = [\overline{\mathrm{P1}} \cdots \overline{P6} \ldots \overline{\mathrm{P11}}] \quad (17)$$

3) GENERATE THE VARIANCE-COVARIANCE MATRIX OF THE DATASET D

The variance-covariance matrix or covariance matrix is computed by establishing a variance relationship between each element of the dataset with the following formula:

$$\mathrm{VC\,(P1, P2)} = \frac{1}{n} \sum_{i=1}^{n} \left(\mathrm{P_{1n}} - \overline{\mathrm{P1}}\right) \left(\mathrm{P_{2n}} - \overline{\mathrm{P2}}\right) \quad (18)$$

The result of the variance-covariance matrix is a square matrix of size p × p; In this research the size of the variance-covariance matrix is 11 × 11. Table 1 is a sample of the variance-covariance space's sake, we illustrate a sample of the matrix with some of the vectors.

4) FIND THE EIGENVALUES AND THEIR EIGENVECTORS

An Eigenvector is in simple terms, a vector which will not change directions after we apply any linear transformation to it [34]. Let us assume our square variance-covariance matrix to be defined by (19).

$$CVM = \begin{pmatrix} VC(P1, P1) & \cdots & VC(P1, P6) & \cdots & VC(P1, P11) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ VC(P6, P1) & \cdots & VC(P6, P6) & \ldots & VC(P6, P11) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ VC(P11, P1) & \ldots & VC(P11, P6) & \ldots & VC(P11, P11) \end{pmatrix} \quad (19)$$

The mathematical expression to find Eigenvalues for the CVM matrix can be presented as follows:

$$\det\left(\mathrm{CVM} - \lambda \mathrm{I}\right) = 0 \quad (20)$$

where λ is the Eigenvalue associated with CVM and I is the identity matrix. An identity matrix corresponding to (19) can be expressed as follows:

$$I = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & \ldots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \ldots & 1 \end{pmatrix} \quad (21)$$

The identity matrix is also a square matrix with the same size as the CVM (11 × 11). Substituting (19) and (21) to (20) and computing the operation results in a eleventh degree equation with λ the unknown. The equation can be represented as:

$$a\lambda^{11} + b\lambda^{10} + h\lambda^{4} + \ldots + i\lambda^{3} + j\lambda^{2} + k\lambda + l = 0 \quad (22)$$

After solving (22), eleven values are found: λ_1, λ_2, λ_3, λ_4, λ_5, λ_6, λ_7, λ_8, λ_9, λ_{10} and λ_{11}; these are the Eigenvalues of our dataset matrix. The next step is to find the corresponding Eigenvectors for each Eigenvalues. Let's assume for each Eigenvalues the following Eigenvectors:

$$\begin{aligned}
\lambda_1 &\rightarrow \mathrm{E_1} = [\mathrm{e_{11}}, \mathrm{e_{12}}, \cdots, \mathrm{e_{111}}] \\
\lambda_2 &\rightarrow \mathrm{E_2} = [\mathrm{e_{21}}, \mathrm{e_{22}}, \cdots, \mathrm{e_{211}}] \\
\lambda_3 &\rightarrow \mathrm{E_3} = [\mathrm{e_{31}}, \mathrm{e_{32}}, \cdots, e_{311}] \\
\lambda_4 &\rightarrow \mathrm{E_4} = [\mathrm{e_{41}}, \mathrm{e_{42}}, \cdots, \mathrm{e_{411}}] \\
\lambda_5 &\rightarrow \mathrm{E_5} = [\mathrm{e_{51}}, \mathrm{e_{52}}, \cdots, \mathrm{e_{511}}] \\
\lambda_6 &\rightarrow \mathrm{E_6} = [\mathrm{e_{61}}, \mathrm{e_{62}}, \cdots, e_{611}] \\
\lambda_7 &\rightarrow \mathrm{E_7} = [\mathrm{e_{71}}, \mathrm{e_{72}}, \ldots, \mathrm{e_{711}}] \\
\lambda_8 &\rightarrow \mathrm{E_8} = [\mathrm{e_{81}}, \mathrm{e_{82}}, \ldots, \mathrm{e_{811}}] \\
\lambda_9 &\rightarrow \mathrm{E_8} = [\mathrm{e_{91}}, \mathrm{e_{92}}, \ldots, \mathrm{e_{911}}] \\
\lambda_{10} &\rightarrow \mathrm{E_{10}} = [\mathrm{e_{101}}, \mathrm{e_{102}}, \ldots, \mathrm{e_{1011}}] \\
\lambda_{11} &\rightarrow \mathrm{E_{11}} = [\mathrm{e_{111}}, \mathrm{e_{112}}, \ldots, \mathrm{e_{1111}}]
\end{aligned}$$

5) REDUCE DATASET DIMENSION BY KEEPING EIGENVECTORS WITH HIGHEST EIGENVALUES

The Eigenvector with the smallest Eigenvalue carry the least information of our data. To effectively reduce the dimension of the dataset we focus on the eigenvectors corresponding to higher Eigenvalues. Since we would like to reduce the size p=11 to a dimension of 2 (2 channels input), we only select the first two higher Eigenvalues and their Eigenvectors. If we consider λ_1 and λ_2 to be the two Eigenvalues with higher values, with $\lambda_1 > \lambda_2$, their corresponding Eigenvectors can be combined into a new matrix (23).

$$G = \begin{pmatrix} e_{11} & e_{12} & \ldots & e_{111} \\ e_{21} & e_{22} & \ldots & e_{211} \end{pmatrix} \quad (23)$$

The reduced dimension of dataset D is computed by the following expression:

$$Z = DG^T \tag{24}$$

where G^T is the transpose of matrix G. G^T is defined in (25):

$$G = \begin{pmatrix} e_{11} & e_{21} \\ e_{12} & e_{22} \\ \vdots & \vdots \\ e_{111} & e_{211} \end{pmatrix} \tag{25}$$

The multiplication of D (14), a 15,000 × 11 matrix, by G^T, a 11 × 2 matrix, by (14) results in new matrix of size 15,000 × 2, with 2 the number of columns of the reduced dataset. The same principal applies for reducing the dimension of any other dataset depending on its size.

B. PARAMETERIZED RECTIFIER LINEAR UNIT (PRELU) ACTIVATION FUNCTION OPTION TO IMPROVE ACCURACY OF LARGER NETWORKS IN CNN MODEL

Improving model performance in machine learning usually implies building more powerful models and designing effective strategies against overfitting. For the past years, several strategies are applied to neural networks to create models more capable of fitting training data: the use of smaller strides [29], [35]–[37], bigger depth [29], [38], new nonlinear activations [39]–[44], enlarged width [35], [36], sophisticated layer designs [38], [45] etc. Researchers introduced some other advanced approaches, such as aggressive data augmentation [10], [29], [38], [46], large-scale data [10], [29], and by effective regularization techniques [44], [47]–[49] to achieve improved generalization. Rectified Linear Unit (ReLU) is one of these approaches under the rectifier neuron [31], [39]–[41] that used to better the success of deep networks [10].

We utilize CNN modeling to perform the classification task for the predictive maintenance framework resources. In this research, we add an optional section to improve the quality of CNN models built for extensive networks by replacing the traditional rectifier linear unit (ReLU) activation function to the Parameterized Rectifier Linear Unit (PReLU). Using PReLU, we improve the model data fitting capability with reduced risk of overfitting when training those models and achieve better accuracy than using traditional ReLU. The PReLU function incorporates ReLU and adds extra parameters that make this technique appropriate for deep networks [50]. As previously mentioned, the PReLU activation function is useful for vast and deep networks in which it improves performance (accuracy). In this research, the improvement is very negligible as working with data of a small manufacturing entity. However, the CNN model we build in this study offers a future proof option for a more substantial amount of data.

Equation (26) is a mathematical expression for an activation function:

$$f(Z_i) = \begin{cases} Z_i & if\ Z_i > 0 \\ c_i Z_i & if\ Z_i \leq 0 \end{cases} \tag{26}$$

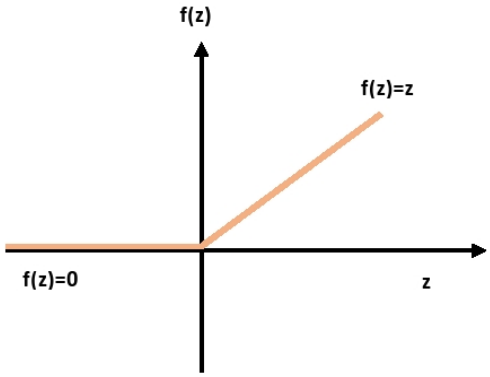

FIGURE 5. ReLU graphical representation.

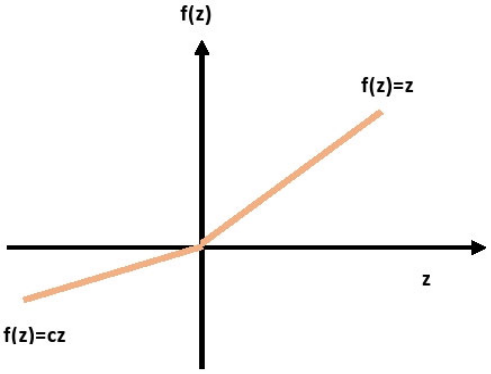

FIGURE 6. PReLU graphical representation.

When c_i is equal to zero, (26) defines a tradition ReLU; when c_i is different than zero or a linear parameter (26) represents PReLU. The graphical representation of (26) for both ReLU and PReLU is presented in Fig.5 and Fig.6 respectively:

From (26), we define Z_i as the input of the activation function f on the *ith* input channel and c_i the coefficient of the slope. (26) can further be expanded as:

$$f(z_i) = \max(0, z_i) + c_i \min(0, z_i) \tag{27}$$

To train PReLU, the back propagation method can be used [51]. Its optimization is done simultaneously for all layers [50]. Equation (28) is used to find the gradient of c_i for one layer:

$$\frac{\partial \varepsilon}{\partial c_i} = \sum_{z_i} \frac{\partial \varepsilon}{\partial f(z_i)} \frac{\partial f(z_i)}{\partial c_i} \tag{28}$$

From (28), ε represents the objection function and $\frac{\partial \varepsilon}{\partial f(z_i)}$ the gradient propagation from the deeper layer of the neural network. To find the gradient of the activation function (29) is used:

$$\frac{\partial f(z_i)}{\partial c_i} = \begin{cases} 0 & if\ z_i > 0 \\ z_i & if\ z_i \leq 0 \end{cases} \tag{29}$$

The next section presents the workflow and steps of our predictive maintenance framework and the main components used for the overall architecture of the system. Elements detailed in previous sections (UTS, MTS, PCA, and CNN

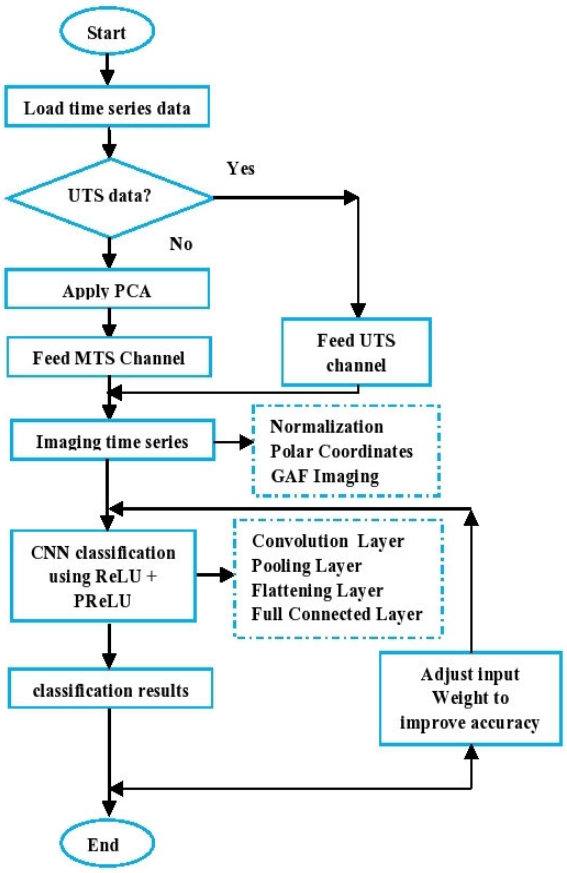

FIGURE 7. Flow chart diagram.

steps) are backbones of the final architecture of the framework.

IV. PREDICTIVE MAINTENANCE FRAMEWORK FLOW DIAGRAM AND OVERALL ARCHITECTURE

The flow diagram that summarizes operations of our experimental predictive maintenance framework is presented in Fig.7. The overall predictive maintenance framework is displayed in Fig.10.

V. EXPERIMENTAL RESULTS

We used data from the conveyor system of a small manufacturing plant to test the effectiveness of our experimental framework. The conveyor system is composed of a conveyor AC motor, a variable frequency drive (VFD), and a conveyor belt with its components. Data preparation is the first step of our experimental framework process. The small manufacturing plant uses to always rely on the motor's vibration speed reading to initiate predictive maintenance on them. Depending on the type of machinery (motor size), vibration thresholds determine their states: normal, warning, or alarm. Fig.8 illustrates the vibration velocity warning and alarm states.

One of the simplest ways to study and understand vibration's signal behavior is to consider it as a sine wave with all its characteristics: amplitude (displacement), period, and

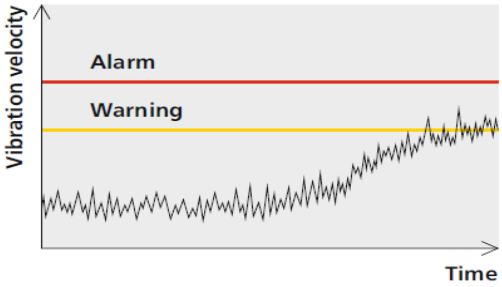

FIGURE 8. Machine vibration trend according to ISO 10816 [52].

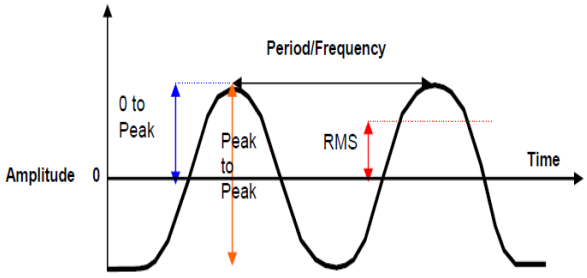

FIGURE 9. Vibration as a sine wave [55].

frequency, as displayed in Fig.9. The velocity or speed of a vibration can is the first derivative of the amplitude, commonly known as the displacement over a certain amount of time or frequency. When analyzing vibrations on their own in systems and equipment, it is not very easy to automatically establish whether the vibrations recorded are harmless to the smooth system's operation. Several rules, such as Table 2, have been developed to address this issue. Another critical practice in this regard is frequent vibration monitoring, useful to detect early impairment in the machinery. The vibration velocity is measured from special vibration sensors (analog signal input) and stored in a controller.

A machine breaks when exposed to several deformations, which is, in reality, a change in amplitude (displacement) that repeatedly occurs at a specific frequency. In other words, the severity of impairment like vibration depends on its displacement and its frequency. As previously mentioned, velocity is also a function of displacement and time (frequency); therefore, the velocity of a vibration can be considered an excellent indicator of vibration severity. For machines operating between 10Hz and 1000Hz, vibration velocity is a good indicator of its severity. For those running at frequencies above 1000Hz, acceleration is one of the most reliable measures of vibration severity [56].

The vibration severity criteria in Table 2 depends on the four classes related to the type and size of the motor used.

Class I is for Small-sized equipment (from 0 to 15KW)

Class II is for Medium-sized equipment (from 15 to 75KW)

Class III is for Large-sized equipment (powered > 75KW) mounted on "Rigid Support" structures and foundations.

Class IV is for Large-sized machines (powered > 75KW) mounted on "Flexible Support" structures [5].

TABLE 2. Vibration severity criteria based on ISO 2372 [5].

RMS Overall Velocity Level in 1000 Hz Bandwidth		Vibration Severity Criteria			
mm/s	**In/s**	**Class I**	**Class II**	**Class III**	**Class IV**
0.28	0.01	Good	Good	Good	Good
0.45	0.02				
0.71	0.03				
1.12	0.04	Satis-factory			
1.8	0.07		Satis-factory		
2.8	0.11	Unsatis-factory		Satis-factory	
4.5	0.18		Unsatis-factory		Satis-factory
7.1	0.28	Unac-ceptable		Unsatis-factory	
11.2	0.44		Unac-ceptable		Unsatis-factory
18	0.71			Unac-ceptable	
28	1.10				Unac-ceptable
45	1.77				

The small manufacturing plant in this experimental research utilizes a class II equipment (medium-sized equipment). From Table 2, all vibration velocity above 4.5mm/s indicates a high severity in the system faults. However, the plant supervisors observed that not all ''critical'' vibration speed would result in a ''critical'' fault and the need to perform predictive maintenance on the system.

Not all apparent ''non-critical'' vibration speeds were safe for conveyor motors since it would cause critical or minor faults. These false falts caused for a long time unnecessary repetitive additional maintenance cost since the existing system would request for a predictive maintenance action while not required or a long-time failure because of misinterpretation of critical faults in need of immediate maintenance action. There was, therefore, a need to take more than one parameter (motor vibration speed) into consideration to accurately detect predictive maintenance schedules. Using specialized sensors (IoT sensors), VFD reading, controllers Inputs/Outputs (I/O) reading, and several other parameters, which combination was observed by supervisors to potentially influenced system failure, were recorded from the conveyor system over a successive period. They are part of the time series dataset and presented in Table 3. For confidentiality reasons, the small manufacturing plant did not disclose their identity and the overall data. The conveyor system operates at 3 main speed controlled by the VFD: $f_1 = 15$Hz, $f_2 = 30$Hz and $f_3 = 50$Hz. The sampling frequency (f_s) used for the experiment is $f_s = 2.56f_3$ that is about $f_s = 128$Hz. Plants operators detected three primary types of faults in the conveyor system:

- **Imbalance:** Unattended broken parts in the conveyor system result in an imbalance in the running machinery that causes vibrations. This fault seems unnoticed at the lowest speed, but it becomes very severe when operating at 30Hz and worst at 50Hz causing vibrations velocity in unsatisfactory range from Table 2 for class II. Undesirable vibrations reduce the accuracy of the conveyor system. Undesirable vibrations reduce the accuracy of the conveyor system. An excessive vibrating conveyor used in a bottling plant spills out content of recipients, stops them on undesired spots, makes them fall on the conveyor and creates congestion in the chain.

- **Misalignment:** Because of the continuous motion of the conveyor system, the machine's shaft gets quickly out of line and causes a misalignment fault that result in undesirable vibrations.

- **Looseness:** In normal conditions, the conveyor system's structure and components need to be stiff and solid to operate smoothly. A decreased in stiffness or loss parts causes vibrations in the system. Vibrations caused by this fault are not very severe and can remain unnoticed without proper monitoring. In the small manufacturing plant, this is the least severe of all three faults. Misalignment or looseness, individually, cause minor fault in the plant (lower vibrations nearing the unsatisfactory border: 4.5 mm/s on Table 2). But when occurring simultaneously result in vibrations velocities in the unsatisfactory spectrum of the vibration severity criteria based on Table 2.

TABLE 3. Time-series dataset variables.

	Parameters	**Units**
Var 1	Vibration speed	m/s
Var 2	Motor Torque	Nm
Var 3	Acceleration	mm^2/s
Var 4	Motor Speed	Hz/s
Var 5	Air Pressure	bar
Var 6	Product Weight	kg
Var 7	Deceleration	mm^2/s
Var 8	Current	A (Amps)
Var 9	Belt tension	N/m
Var 10	Motor tension	N/m
Var 11	Temperature	*C

Parameters in Table 3 represent the independent variables of our deep learning model. A combination of these variables determines three states of the conveyor motor: (1) No-Fault (2) Minor Fault and (3) Critical Fault with urgent need of maintenance. or confidentiality sake, we display, in Table 4(in the appendix section), only one combination sample for each state. Fig.11 presents the time series variables from Table 3. We selected a portion of the dataset over a smaller period for visibility purposes. There is a need for data conversion and scaling in controllers and Supervisory Control And Data Acquisition (SCADA) for field operators to interpret information easily.

This research built a classification model that studies in depth the combination of all the above parameters and generates a more accurate way to detect critical faults in need of immediate maintenance, minors faults more negligible, and no faults. From our dataset, we can tell that we are dealing with MTS input data. Therefore, as per our framework

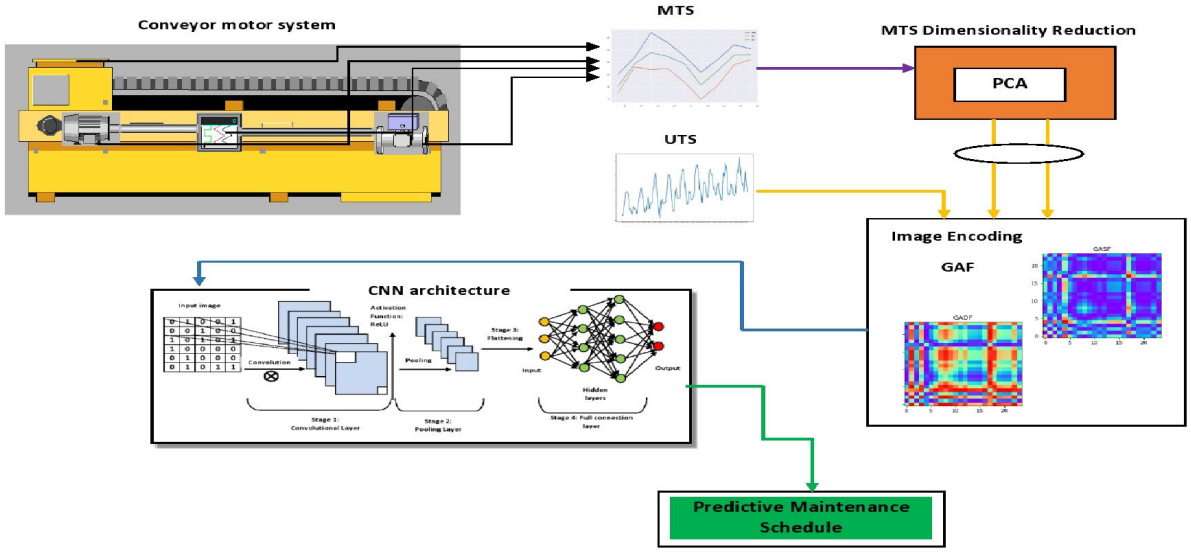

FIGURE 10. Overall system's architecture.

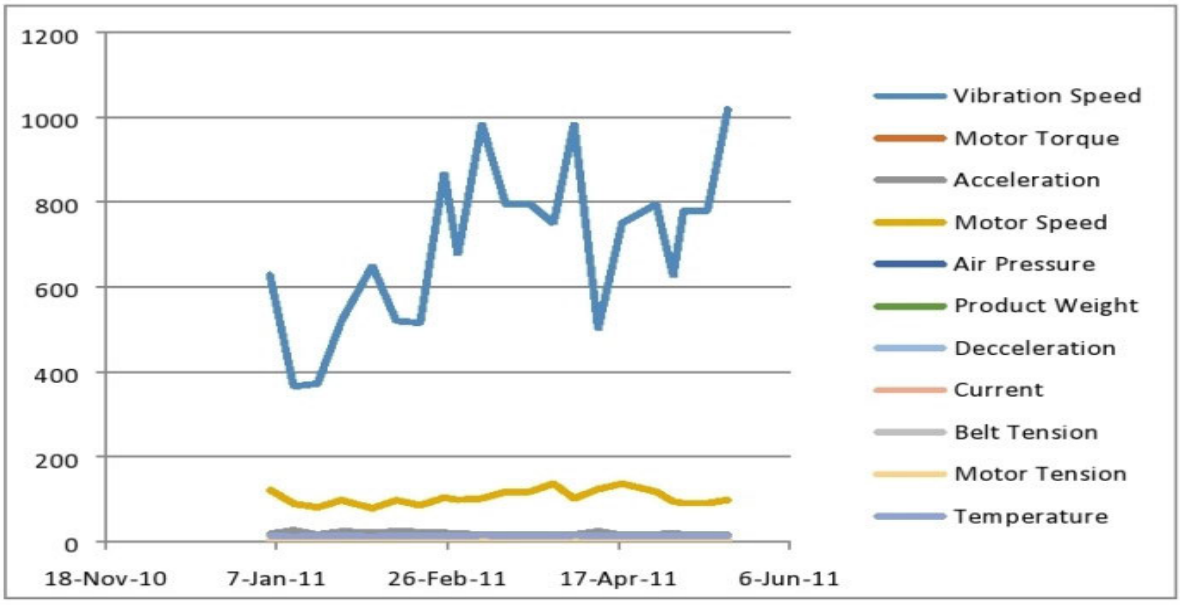

FIGURE 11. Independent variables MTS plot.

workflow, we apply PCA to reduce the dimension of the independent variables to a maximum of two channels.

A. ALGORITHM SETTINGS SUMMARY: PCA

Table 4 is a summary of important settings used for the dimensionality reduction of the MTS to PCA (The 'R' platform was used for PCA dimensionality reduction):

PCA algorithm generates two new sets of independent variables replacing all variables in Table 3. The two variables are named PCAvar1 and PCAvar2, with their values different from original raw data. Fig.12 is a graphical representation of these two variables. We used the same time interval for both Fig.11 and Fig.12.

TABLE 4. PCA settings.

Settings	Value	Comment
SET.SEED	123	
SPLIT RATIO	0.8	Training and Test Set
METHOD	PCA	
PCACOMP	2	2 Channels used for MTS

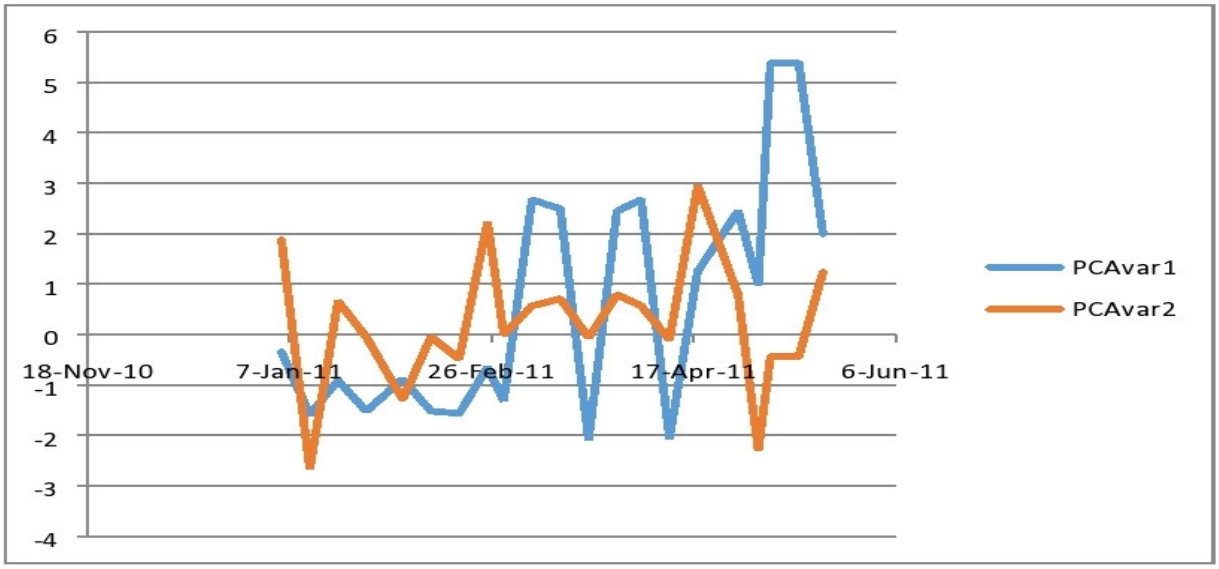

FIGURE 12. PCA variables representing reduced independent MTS variables.

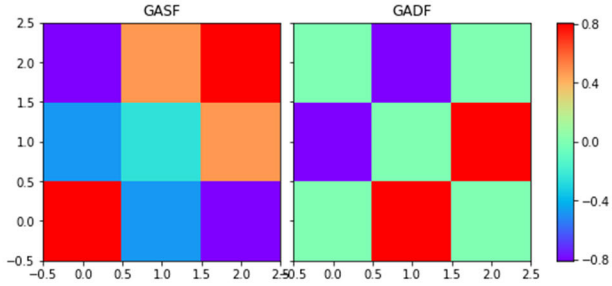

FIGURE 13. "No Fault (NF)" motor status sample on GAF images.

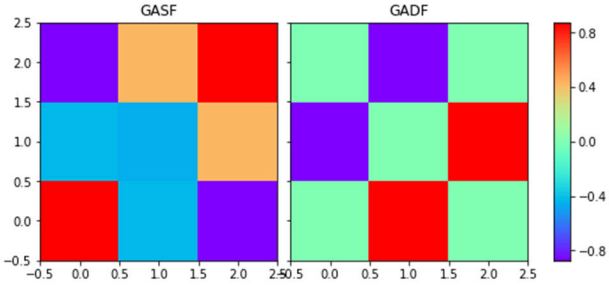

FIGURE 14. "Minor Fault (MF)" motor status sample on GAF images.

B. ALGORITHM SETTINGS SUMMARY: GAF

We load the new independent variables from Fig.12 into the image encoding section where they are (1) normalized, (2) converted to polar coordinates, and (3) transformed to GAF images. We used the python platform to generate GAF images for each case of the dataset. The following are the most critical settings and steps for this section:

- Gramian angular field library imported from pyts.image.
- Separate each motor condition cases from PCA variables.
- Load each three cases ('1', '2' and '3') individually in the GAF code.
- Image_Size: 3
- Generate and save "summation" (GASF) and "difference" (GADF) images for each line data: from X_gasf [0], X_gadf [0] to X_gasf[n], X_gadf[n] (n is the last item number of each case)
- Save images in separate folders based on cases.

Fig.13, Fig.14, and Fig.15 are the image samples of the three motor states are No-fault ('3'), Minor fault ('2'), and Critical fault ('1'), classified by our framework.

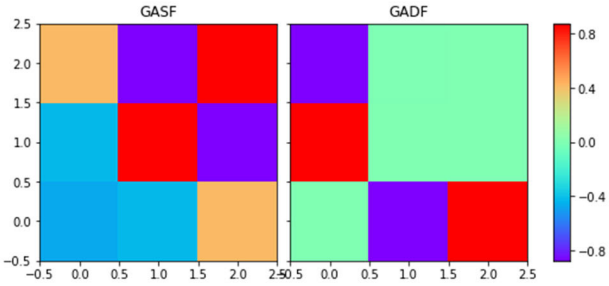

FIGURE 15. "Critical Fault (CF)" motor status sample on GAF images.

Most critical faults in the system were caused by an imbalance system when running the system at 50Hz. Some others were caused by a combination of at least two faults, for example, misalignment and looseness occurring at the same time. The minor fault was usually a result of misalignment or looseness individually.

Our research aims to build an effective classification model that uses several motor parameters and observations as inputs of the system. These inputs are:

TABLE 5. SVM settings.

Settings	Value
Function	SVM
Type	C-Classification
Kernel	Linear

- Vibration speed (VS),
- Motor torque (MT),
- Acceleration (ACC),
- Motor speed (MS),
- Air pressure (AP),
- Product weight (PW),
- Deceleration (DEC),
- Current(CUR),
- Belt tension (BT),
- Motor tension (MT-S)
- Temperature (TMP)

The output of the model is the ''Fault Severity in the System'' that determines a predictive maintenance schedule. The fault severity or the output of the system has three conditions displayed in Fig.13, Fig.14, and Fig.15: No-Fault, Minor Fault, and Critical Fault (that requires immediate action), respectively. Fig.16 is a representation of our predictive framework inputs/Outputs (I/O) model.

C. ALGORITHM SETTINGS SUMMARY: SVM AND CNN

As previously mentioned, our framework machine learning section offers an optional section for more extensive networks by building a CNN model based on the PReLU activation function instead of the standard ReLU. To evaluate our results, we use three machine learning models:(1) Support Vector Machine (SVM), (2) Standard CNN (using ReLU), and (3) CNN + PReLU.

The SVM model is built in an R platform using PCA variables. The critical parameters used to generate are:

The CNN models are built in a python platform using images generated in GAF and saved in different folders. We divided all converted images into two categories: a training set and a test set. The training set applied to train and build our CNN models and the test set to measure the accuracy of the model. This step is the pre-processing data phase of modeling our CNNs. Unlike SVM models, where all this is done automatically in the machine learning script, this part is achieved manually for CNN in this study.

Parameters settings in Table 6 are worth mentioning for the CNN models.

Dealing with three classification models, the evaluation metrics used for the above models are:

- Accuracy: The accuracy of a classification model can be defined as the percentage of correct predictions of the overall model over the total number of samples used for prediction. Let assume CP the number of correct predictions in a model and n the total number of instances used in the test set to evaluate the model, the accuracy can be represented by (30) as follows:

$$accuracy = \frac{CP}{n} 100\% \tag{30}$$

TABLE 6. CNN parameters settings.

Parameters	Value	Comment
Split ratio	0.8	This corresponds to about 12,000 images in the training set and 3,000 images in the test or validation set
Library	Keras and tensorFLOW BACKEND	
Feature detectors size	(32, 3 , 3)	Since using a normal Central Processing Unit (CPU) instead of Graphical Processing Unit (GPU) we reduce the feature detector size to 32 to have less processing time.
Input shape	(64, 64, 3)	Since using a normal CPU instead of GPU we reduce the input shape to 64 to have less processing time.
Activation function	- ReLU (CNN + ReLU model), - PReLU (with alpha initializer = 0) for CNN+PReLU.	
Pool size under max pooler	(2, 2)	
Output dimension	128	Number of hidden nodes in the full connection section first layer
Output dimension	3	In the full connection last layer
Action function in the last layer	Softmax	
Compiling the model	Optimizer: "Adam" Loss: "categorical_crossentropy" Metrics: "Accuracy"	
Insert image data Generator to increase data size		
Extracting the training set images:	Target size: (64,64) Batch size: 32 Class mode: Categorical (3 classes to predict)	
Extracting the test (validation) set images:	Target size: (64,64) Batch size: 32 Class mode: Categorical **Shuffle: False**	
Fitting the CNN model to training set and testing using the test set	Samples per epoch: 12,000; Number of Epochs: 3, number of validation samples: 3,000.	

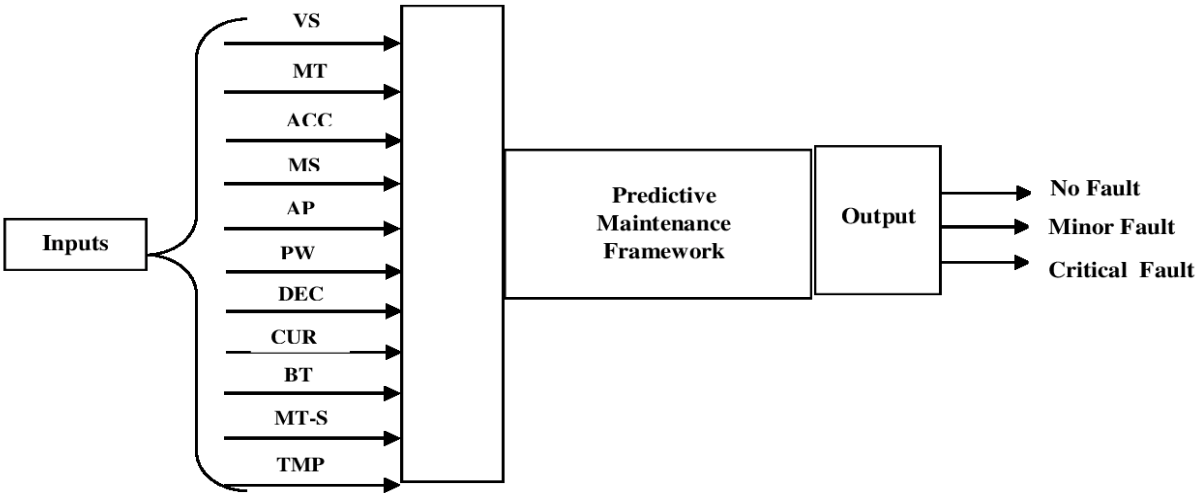

FIGURE 16. Framework Inputs/Outputs architecture.

- Precision: Another evaluation metric is precision defined as the percentage of correct prediction for each different class, individually, over the total number of instances predicted for those classes. In this research our dataset has three classes for the classification model: no fault (NF), minor fault (MF) and critical fault (CF). Equation (31) is the mathematical expression for the precision:

$$precision = \frac{CP_m}{P_m}100\%, \quad m \in \mathbb{N} \tag{31}$$

where m is the number of classes of the dataset, CP_m is the number of correct predictions per class, p_m is the total number of instances predicted for that class (correct and incorrect).

- Recall: The recall is known as the percentage of instances of a class that were correctly predicted. In other words it is a ratio of the number of correct predictions of a class over the sum of correct predictions and missed correct predictions. Its mathematical expression is presented on (32).

$$recall = \frac{CP_m}{(CP_m + IP_m)}100\%, \quad m \in \mathbb{N} \tag{32}$$

where m is the number of classes of the dataset, CP_m is the number of correct predictions per class, p_m is the total number of instances predicted for that class (correct and incorrect).

While accuracy is an evaluation metric for the overall model (all classes included), Precision and Recall are useful to have insights on individual classes and interpret the behavior of each class better.

A confusion matrix is an essential tool that displays a summary of classification results, mainly the actual labels versus predicted ones [53]. It also computes the accuracy, precision, and recall of a classification model. Tables 7, 8, and 9 are confusion matrices of the three experimental classification models used in our predictive maintenance framework.

On the above confusion matrices, the green-colored cells are the number of correct predictions made by the model for each class. The remaining uncolored cells contain the number of incorrect predictions. The meaning of the labels is (1) CF: critical fault, (2) MF: minor fault, and (3) NF: no-fault. We reduced the dimension of the input data for all three machine learning models by applying PCA.

TABLE 7. Confusion matrix result of SVM model.

	CF	MF	NF
CF	612	169	224
MF	149	181	706
NF	0	96	864

TABLE 8. Confusion matrix result of CNN + RELU.

	CF	MF	NF
CF	1000	0	0
MF	0	1000	0
NF	0	0	1000

TABLE 9. Confusion matrix result of CNN + PRELU.

	CF	MF	NF
CF	1000	0	0
MF	0	1000	0
NF	0	0	1000

Note: Confusion matrices results of both CNN models are quite impressive with 100% positive predictions. We achieved an outstanding prediction by running three epochs, which is the number of times the CNN algorithm learns the model behavior using available training set data when the training and testing of these models. Training and

TABLE 10. Classification models results summary.

Classification Models	Label	Precision	Recall	Overall Accuracy
SVM	CF	0.8042	0.6090	0.552≈ 55.2%
	MF	0.4058	0.1747	
	NF	0.4816	0.9000	
CNN+ ReLU	CF	0.1000	0.1000	0.100≈ 100.0%
	MF	0.1000	0.1000	
	NF	0.1000	0.1000	
CNN + PReLU	CF	0.1000	0.1000	0.100≈ 100.0%
	MF	0.1000	0.1000	
	NF	0.1000	0.1000	

validation (testing) accuracies obtained at the last epoch for both models are very close to just less than 1% (99, xx% - 100%), which reflects models with fewer chances of overfitting. Table 7 summarizes the evaluation metrics results of the three classification models used in this research:

As per the results in Table 10, using CNN for predictive maintenance increases the experimental system's accuracy to almost 50% as opposed to utilizing a traditional SVM machine learning model. Although the preparation and modeling steps of a predictive maintenance system using CNN may seem tedious and demanding, the hardest part of the modeling is done once in the beginning. The remaining operations will be a fine-tuning of parameters and loading new observations in the system for the algorithm to improve its performance. Depending on the plant activities, the supervisors can perform this operation once a month or during maintenance and shutdown. Results obtained for both CNN+ReLU and CNN+PReLU are identical for relatively small datasets but could make a difference for more extensive networks. This option is, therefore, quite handy for a future proof model, which will undoubtedly have to deal with a more significant number of data. The best overall accuracy achieved is 100%. These are outstanding results that bring more effectiveness and reliability in the system and makes a big difference when predicting machine conditions (conveyor motors) since an incorrect prediction could either result in critical breakdown or unnecessary maintenance expenses. As the production/manufacturing systems are different, it is essential to conduct a proper study on each system behavior before implementing the adequate predictive maintenance approach.

VI. CONCLUSION

This research presents an experimental framework that triggers effective predictive maintenance for conveyor motors in a small manufacturing industry utilizing a classification model built through time-series input data imaging and CNN with a future-proof option for more extensive networks using PReLU activation function to improve model performance. Several conveyor system parameters observed sequentially are converted into images using GASF that has the advantage of preserving temporal features. Experimental results obtained show the relevance of using deep learning algorithms such as CNN to improve the accuracy of classification models. Our predictive maintenance framework architecture accommodates both UTS and MTS data input. It classifies the conveyor motor status into three categories based on input parameters: critical fault, minor fault, and no-fault. The best overall accuracy achieved by our experimental framework is about 100%, which is quite sufficient for initiating predicting maintenance schedules. With these excellent results, our framework reduces the high risk of missing critical faults in the system, which could lead to a more prolonged breakdown or unnecessarily initiating maintenance on motors due to incorrect predictions leading to a waste of resources.

For future work, we would like to expend the framework's ability to deal with diverse data types by adding a feature that includes non-linear time series input. We could reduce the dimensionality of the non-linear data through the Kernel PCA algorithm; we would also fine-tune our CNN models by incorporating additional parameters such as "Dropout," which can prevent the risk of overfitting. Furthermore, we would like to incorporate CNN classifications results of the predictive maintenance framework in the operational technology (OT) environment where we include classification motor statuses in Supervisory Control And Data Acquisition (SCADA) displayed on a Human Machine Interface (HMI) and remotely accessible in cloud-based applications by supervisors and operators.

REFERENCES

[1] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting," *Electronics*, vol. 8, no. 8, p. 876, Aug. 2019.

[2] R. Zhang, F. Zheng, and W. Min, "Sequential behavioral data processing using deep learning and the Markov transition field in online fraud detection," *Math., Comput. Sci.*, vol. 1808, no. 05329, pp. 1–5, Aug. 2018. [Online]. Available: https://arxiv.org/pdf/1808.05329.pdf

[3] C. L. Yang, Z. X. Chen, and C. Y. Yang, "Sensor classification using convolutional neural network by encoding multivariate time series as two-dimensional colored image," *Sensors*, vol. 20, no. 1, 168, pp. 1–15, Dec. 2019, doi: 10.3390/s20010168.

[4] M. Gahirwal and M. Vijayalakshmi, "Inter time series sales forecasting," *Int. Journ. Adv. Stud. Comp. Sci. Eng.*, vol. 2, no. 1, pp. 55–66, Mar. 2013. [Online]. Available: https://arxiv.org/pdf/1303.0117.pdf

[5] K. S. Kiangala and Z. Wang, "Initiating predictive maintenance for a conveyor motor in a bottling plant using industry 4.0 concepts," *Int. J. Adv. Manuf. Technol.*, vol. 97, nos. 9–12, pp. 3251–3271, May 2018.

[6] K. Wang, "Intelligent predictive maintenance (IPdM) system-industry 4.0 scenario," *WIT Trans. Eng. Sci.*, vol. 113, no. 10, pp. 259–268, 2016.

[7] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). New York, NY, USA: Springer-Verlag, 2006.

[9] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25. Red Hook, NY, USA: Curran Associates, Dec. 2012, pp. 1097–1105.

[11] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," 2015, *arXiv:1506.00327*. [Online]. Available: http://arxiv.org/abs/1506.00327

[12] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Proc. AAAI Workshops*, Jan. 2015, pp. 40–46.

[13] W. Chen and K. Shi, "A deep learning framework for time series classification using relative position matrix and convolutional neural network," *Neurocomputing*, vol. 359, pp. 384–394, Sep. 2019.
[14] G. Martínez-Arellano, G. Terrazas, and S. Ratchev, "Tool wear classification using time series imaging and deep learning," *Int. J. Adv. Manuf. Technol.*, vol. 104, nos. 9–12, pp. 3647–3662, Oct. 2019.
[15] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," 2013, *arXiv:1302.6613*. [Online]. Available: http://arxiv.org/abs/1302.6613
[16] L. Sadouk, "CNN approaches for time series classification," in *Time Series Analysis Methods and Applications for Flight Data*. London, U.K.: IntechOpen, 2018. [Online]. Available: https://www.intechopen.com/books/time-series-analysis-data-methods-and-applications/cnn-approaches-for-time-series-classification, doi: 10.5772/intechopen.81170.
[17] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. New York, NY, USA: Wiley, 2015.
[18] T. Górecki and M. Łuczak, "Multivariate time series classification with parametric derivative dynamic time warping," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2305–2312, Apr. 2015.
[19] A. Coraddu, L. Oneto, A. Ghio, S. Savio, D. Anguita, and M. Figari, "Machine learning approaches for improving condition-based maintenance of naval propulsion plants," *Proc. Inst. Mech. Eng., Part M, J. Eng. Maritime Environ.*, vol. 230, no. 1, pp. 136–153, Feb. 2016.
[20] K. Leahy, R. L. Hu, I. C. Konstantakopoulos, C. J. Spanos, and A. M. Agogino, "Diagnosing wind turbine faults using machine learning techniques applied to operational data," in *Proc. IEEE Int. Conf. Prognostics Health Manage. (ICPHM)*, Jun. 2016, pp. 1–8.
[21] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine learning for predictive maintenance: A multiple classifier approach," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 812–820, Jun. 2015.
[22] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, "Machine learning approach for predictive maintenance in industry 4.0," in *Proc. 14th IEEE/ASME Int. Conf. Mech. Embedded Syst. Appl. (MESA)*, Jul. 2018, pp. 1–6.
[23] K. Kulkarni, U. Devi, A. Singhee, J. Hazra, and P. Rao, "Predictive maintenance for supermarket refrigeration systems using only case temperature data," in *Proc. Ann. Amer. Contr. Conf. (ACC)*, Milwaukee, WI, USA, Jun. 2018, pp. 4640–4645.
[24] J. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.
[25] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.
[26] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*. Sebastopol, CA, USA: O'Reilly Media, 2017.
[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst. Conf.*, 1990, pp. 396–404.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*. [Online]. Available: http://arxiv.org/abs/1512.03385
[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015, *arXiv:1512.00567*. [Online]. Available: http://arxiv.org/abs/1512.00567
[31] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[32] Y. T. Zhou and R. Chellappa, "Computation of optical flow using a neural network," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 2, Jul. 1988, pp. 71–78.
[33] A. Devarakonda, M. Naumov, and M. Garland, "AdaBatch: Adaptive batch sizes for training deep neural networks," 2017, *arXiv:1712.02029*. [Online]. Available: http://arxiv.org/abs/1712.02029
[34] M. Munir, "Eigenvalues-theory and applications," Dept. Math., Govern. Postgrad. College, Karl-Franzens-Universität, Graz, Austria, Tech. Rep., 2015. [Online]. Available: https://www.researchgate.net/publication/309012418_Eigenvalues-Theory_and_Applications, doi: 10.13140/RG.2.2.15926.91201.
[35] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[36] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, Scottsdale, AZ, USA, 2014, pp. 1–16.
[37] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," 2014, *arXiv:1405.3531*. [Online]. Available: http://arxiv.org/abs/1405.3531
[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014, *arXiv:1409.4842*. [Online]. Available: http://arxiv.org/abs/1409.4842
[39] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.
[40] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.(ICML)*, 2013, pp. 1–6.
[41] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Vancouver, BC, Canada, May 2013, pp. 3517–3521.
[42] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: http://arxiv.org/abs/1312.4400
[43] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, "Compete to compute," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2013, pp. 2310–2318.
[44] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 28, no. 3, Jun. 2013, pp. 1319–1327.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[46] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," 2013, *arXiv:1312.5402*. [Online]. Available: http://arxiv.org/abs/1312.5402
[47] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*. [Online]. Available: http://arxiv.org/abs/1207.0580
[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[49] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.
[50] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[51] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
[52] IFM Electronics. (2013). *From Process Monitoring to Vibration Analysis*. Condition Monitoring Systems. pp. 5–12. [Online]. Available: http://www.ifm.com/download/files/ifm-efector-octavis-brochure-GB-2013/$file/ifm-efector-octavis-brochure-GB-2013.pdf
[53] S. Visa, B. Ramsay, A. L. Ralescu, and E. Van Der Knaap, "Confusion matrix-based feature selection," in *Proc. Midwest Artif. Intel. Cognit. Scienc. Conf.*, vol. 710, 2011, pp. 120–127.
[54] A. Yunusa-Kaltungo, J. K. Sinha, and A. D. Nembhard, "A novel fault diagnosis technique for enhancing maintenance and reliability of rotating machines," *Struct. Health Monit., Int. J.*, vol. 14, no. 6, pp. 604–621, Nov. 2015, doi: 10.1177/1475921715604388.
[55] C. Sanders. (2011). A guide to vibration analysis and associated techniques in condition monitoring. DAK Consulting-Chiltern House. [Online]. Available: http://www.dakacademy.com/newsite/index.php?option=com_k2&Itemid=500&id=94_007cd4b8b347e375bc10dbe5efbccc28&lang=en&task=download&view=item
[56] J. Alsalaet. (Dec. 2012). *Vibration Analysis and Diagnostic Guide*. [Online]. Available: https://www.researchgate.net/publication/311420765_Vibration_Analysis_and_Diagnostic_Guide

KAHIOMBA SONIA KIANGALA received the B.Tech. degree in electrical engineering from the Tshwane University of Technology (TUT), South Africa, in 2014, and the master's degree in electrical engineering from the University of South Africa (UNISA), in 2019, where she is currently pursuing the Ph.D. degree in science, engineering and technology (SET). Her research interest includes adapting artificial intelligence techniques to improve automation techniques for small to medium scale industries in an Industry 4.0 environment.

ZENGHUI WANG (Member, IEEE) received the B.Eng. degree in automation from the Naval Aviation Engineering Academy, China, in 2002, and the Ph.D. degree in control theory and control engineering from Nankai University, China, in 2007. He is currently a Professor with the Department of Electrical and Mining Engineering, University of South Africa (UNISA), South Africa. His research interests are industry 4.0, control theory and control engineering, engineering optimization, image/video processing, artificial intelligence, and chaos.