INTERNATIONAL JOURNAL OF AUTOMOTIVE AND MECHANICAL ENGINEERING (IJAME)
ISSN: 2229-8649 e-ISSN: 2180-1606
VOL. 19, ISSUE 2, 9668 – 9680
DOI: https://doi.org/10.15282/ijame.19.2.2022.04.0746

ORIGINAL ARTICLE

Unbalance Failure Recognition Using Recurrent Neural Network

M.M. Ruslan and M.F. Hassan*

College of Engineering, Universiti Malaysia Pahang, Kuantan, 26300 Pahang, Malaysia.

ABSTRACT – Many machine learning models have been created in recent years, which focus on recognising bearings and gearboxes with less attention on detecting unbalance issues. Unbalance is a fundamental issue that frequently occurs in deteriorating machinery, which requires checking prior to significant faults such as bearing and gearbox failures. Unbalance will propagate unless correction happens, causing damage to neighbouring components, such as bearings and mechanical seals. Because recurrent neural networks are well-known for their performance with sequential data, in this study, RNN is proposed to be developed using only two statistical moments known as the crest factor and kurtosis, with the goal of producing an RNN capable of producing better unbalanced fault predictions than existing machine learning models. The results reveal that RNN prediction efficacies are dependent on how the input data is prepared, with separate datasets of unbalanced data producing more accurate predictions than bulk datasets and combined datasets. This study shows that if the dataset is prepared in a specific way, RNN has a stronger prediction capability, and a future study will explore a new parameter to be fused along with present statistical moments to increase RNN's prediction capability.

ARTICLE HISTORY
Received: 11th Nov 2021
Revised: 6th Jan 2022
Accepted: 30th Mar 2022
Published: 28th June 2022

KEYWORDS
Recurrent neural network;
Vibration;
Unbalance;
Fault prediction

INTRODUCTION

Fault prediction is a critical component of a successful operation in any business, but it is especially important in the energy industry because of its ongoing demands. Most heavy industries, such as oil and gas, energy and power plants, and chemical plants, adopt condition-based maintenance (CBM), in which the equipment is monitored on a regular basis and parts are replaced only when deteriorating tendencies are detected using condition monitoring methods. Vibration analysis is one of the methods that have been shown to improve machine dependability and reduce breakdown maintenance [1].

Because the recent pandemic outbreak has had a significant impact on energy prices, oil corporations have chosen to reduce costs by implementing integration and automation [2]. Smart devices enable automation, where technology such as remote monitoring which operates through sensor applications that work over wired and/or wireless connections, comes into play and manages to reduce the number of personnel on the field performing costly and time-consuming visual inspections [3, 4]. Smart pigs with transmitters, sensors, Global Positioning System (GPS), eddy current, magnetic fields, ultrasonics, and acoustics could help to reduce this. Aba et al. conducted a study that demonstrated that data visualisation via wireless communication can achieve a satisfactory level when used for real-time monitoring [4], in which they used vibration-based methods that used time-delay methods between pressure pulse arrivals to detect damaged locations on pipelines. In the oil and gas industry, the usage of data recording devices has resulted in large databases [5].

In recent years, machine learning (ML) for machinery fault prediction has piqued the interest of many academics. Because of the abundance of data that exists as a byproduct of online monitoring, where data is continuously watched and saved on a server or computer hard drive, this data-driven method has gained traction. This has given scholars the opportunity to use them as training materials for machine learning to learn about the many machinery situations found in historical data. As a result, a range of machine learning algorithms have been created, some of which are geared to forecast equipment failures, while others estimate the remaining useful life (RUL) of the machinery over its lifetime [6]. ML has been the go-to alternative for performing vibration-based fault diagnosis due to its nature, which operates without requiring a high level of expertise as traditional methodologies do [7]. Many scholars use the bearing data published by Case Western Reserve University (CWRU) on their data centre website [8] to develop ML models capable of recognising bearing faults, such as Deep Recurrent Neural Networks, Multivariable Ensemble-based Incremental Support Vector Machine, and Long Short-Term Memory [9, 10, 11]. ML methods, such as artificial neural networks and convolutional neural networks, were built using bearing datasets published by NASA [12] and gearbox datasets collected from the data bank of the Prognostics and Health Management (PHM) society's annual competition [13]. As the field of intelligent bearing diagnosis develops, there are very few studies that have been done to develop machine learning for unbalanced fault detection, making it difficult to find datasets containing unbalanced problems online. However, because Oliver Mey and colleagues used Convolutional Neural Networks (CNNs), Fully-Connected Neural Networks (FCNNs), Random Forests, and Hidden Markov Models to detect unbalance issues in rotating machinery, their data has been made available [14].

*CORRESPONDING AUTHOR | M. F. Hassan | ✉ firdaus@ump.edu.my

Bearings have been demonstrated to play vital roles in rotating machinery such as electric motors, turbines, pumps, and other machines that would be prone to catastrophic failure if not maintained properly. Bearing issues scored as the highest tier (40%) among other defects in electric motors, according to Sharma et al. [16]. Other faults included stator winding faults (35%), rotor faults (15%), and other faults combined (10%). While bearing failures are the most common cause of machine failure, many people ignore other issues such as unbalance. Unbalance, in contrast to bearing fault, is considered a fundamental problem since it can be recognised at audible frequencies, which are lower than the early bearing fault frequencies, which are found in ultrasound ranges of around 20 kHz [13]. Because recording its characteristics does not necessitate particular settings inside the data acquisition device, looking for unbalance does not require in-depth vibration knowledge. Unbalance in rotating machinery that is not addressed can spread and harm adjacent components, reducing their remaining useful life dramatically. Critical unbalance could cause the system to generate strong vibration unbalance, which could lead to rotor breakage, bearing wear, and other failures [17], as well as a loud noise that could impact one's hearing abilities.

Rampant vibration can result in torsional vibration and electromagnetic flux fluctuation, which are particularly problematic in electric motors. Torsional vibration causes the motor speed to fluctuate, which prematurely deteriorates the mechanical coupling, while the latter damages the electrical winding [18]. As a result of these examples, unbalance should not be taken lightly and should be addressed before it has a negative impact on the rest of the system and increases maintenance costs.

When it comes to machinery prognosis that involves vibration signals, the quality of the characteristics retrieved from the vibration data plays a big part in recognising issues. Meng et al. proposed shaft orbit as a support feature to the vibration energy feature in their study [19] and were successful in creating a Support Vector Machine (SVM) to detect rotor unbalance. Additionally, one of the elements that must be considered is the amount of data that must be put into the machine learning algorithm. In the case of limited data availability, Gangsar et al. have demonstrated that an SVM can detect unbalance within an induction motor with abundant data extraction from the first statistical moment to the extreme of the eighth statistical moment, providing an ample amount of information for the SVM to learn in detecting unbalance [20]. There have been studies on performing unbalance fault recognition by fusing datasets to be learned by machine learning models in two ways: the first is through the combination of datasets of unbalanced faults from two different intensities, both of which were extracted from vibration signals, as carried out by Mey et al. [15], and the second is through the combination of vibration signals and shaft orbit information, as carried out by Yan et al. in training their Deep Belief Network (DBN) in recognising rotor unbalance fault [21]. The former has unbalanced fault recognition accuracy that is lower than non-combination datasets-trained models, while the latter has prediction accuracy that is higher than non-combination dataset-trained black boxes.

Because there are so few scholars working on developing machine learning models to detect unbalanced faults, datasets containing unbalance faults are hard to come by. Recurrent Neural Networks (RNNs) have been widely used to analyse data that is temporal or time-related, such as voice and vibration data [22, 23], because of their ability to produce highly accurate data prediction by using previous output as feedback to fresh input. RNN has not only been shown to be practical and accurate in performing natural language processing (NLP) and time serial prediction, but it may also be the most powerful method in prediction, including but not limited to fault prognosis and speech recognition [11, 10], as well as capturing the time relevance of data in a time sequence. For example, Liu et al. found that RNN produced a 6 % relative improvement in voice recognition when compared to Time Delay Neural Network (TDNN) [22]. In addition, Xie et al. presented a study that performed prognosis on the future condition of worsening bearing state, which was compared to the findings obtained by Support Vector Machine (SVM) and Echo State Network (ESN), with RNN outperforming the two [11]. Using RNN in the form of an autoencoder, it is also possible to anticipate the future period based on the preceding period of bearing fault. In comparison to what was produced by SVM, auto-encoder (AE), and denoising autoencoder (DE) in the experiment, the proposed approaches demonstrated satisfactory performance with high classification accuracy.

Since no study involving unbalance prediction has ever used RNN as their model, writers have determined that RNN is the correct model to employ to forecast unbalance fault based on the cited studies. As a result, the use of datasets by Mey et al. to assess the performances of their CNNs and FCNNs in forecasting unbalance has led this work to creating many RNN models using a range of architectural designs. The following sections explain the way RNN is crafted in order to investigate the use of RNN as the main model to predict unbalance faults based on published online data, the effect of open-loop and close loop for the RNN training stage on prediction accuracy, and lastly, the impact of retraining on black box prediction capability.

MACHINE LEARNING

Machine learning is a method of utilising artificial intelligence to learn patterns of data, usually for the purpose of deriving prediction models without human aid and without requiring the user to understand the details of the underlying codes behind them [24]. Thus, the trained ML model is regarded as a 'black box' due to its complex behaviour that is difficult to understand. The development of RNN is introduced in this section, beginning with RNN initial architecture development as suggested in the MathWorks® documentation, which was later tailored based on the produced results to achieve a more reliable fault prediction capability.

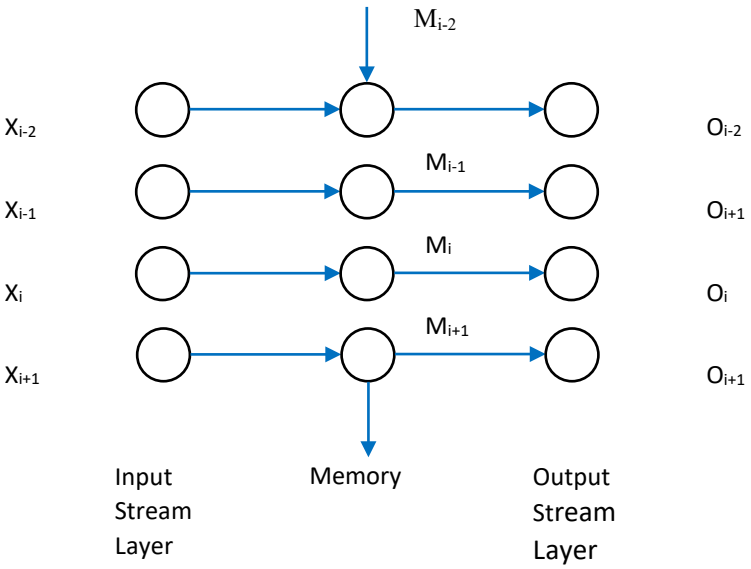


Figure 1. Recurrent Neural Network [24].

Recurrent Neural Network

RNN is deemed as one of the ML models that is suitable for sequential data analysis since its core data analysis tasks are executed by hidden units that are capable of keeping information about the past steps. As depicted in Figure 1, the output at t1 will affect the parameter at t1 + 1. Due to this, RNN utilises information from the past and the present to come up with new data, which allows it to perform predictions better than other ML models [27]. Due to the nature of vibration of rotating machinery that is sequential and changes throughout time as the condition worsens, RNN is regarded as a suitable proposition to deal with stochastic behaviour within long sequential datasets.

PROPOSED METHOD

This section discusses steps within RNN constructions which include the architectures, parameters extraction, training stages, and prediction efficacies.

Parameters Preparation

In this study, the works and datasets of Oliver Mey and his colleagues in creating ML models for unbalanced predictions are used as benchmarks, their published datasets were taken for extraction to produce inputs to the RNN in the training stage. Table 1 and 2 show the description of Mey et al.'s datasets of unbalance faults that were collected from their test rig as per the depicted in page 1610 of ref. [20]. The test rig was made of a DC motor whose speed was regulated by a motor controller that enabled it to run at a speed of 300 to 2,300 revolutions per minute (RPM). The motor shaft was then connected to the 75 mm shaft via a coupling, where the latter was held by a bearing within a pedestal block made of galvanised steel. The unbalance disc, which was made through additive manufacturing, uses nylon as the main material. The disc was fabricated at 52 mm in diameter and designed with axially symmetrical recesses for weight insertion to simulate unbalances. Three vibration sensors were used to acquire vibration signals from the motor and bearing, of which the former was fitted with one accelerometer in the vertical direction, while the latter was fitted with the remaining accelerometers in the vertical and horizontal directions at the bearing pedestal.

Table 1. Unbalance datasets for training by Mey et al. [15].

Identity	Radius (mm)	Mass (g)	Unbalance factor (mm.g)	Number of samples	
				Development (D) (training)	Total number of data
0D	-	0	0	6,438	26,370,048
1D	14 ± 0.1	3.281 ± 0.003	45.9 ± 1.4	6,434	26,353,664
2D	18.5 ± 0.1	3.281 ± 0.003	60.7 ± 1.9	6,434	26,353,664
3D	23 ± 0.1	3.281 ± 0.003	75.5 ± 2.3	6,430	26,337,280
4D	23 ± 0.1	6.614 ± 0.007	152.1 ± 2.3	6,430	26,337,280

The data was recorded accordingly based on unbalance intensity in an incremental manner, starting from 0 towards 4, where 0 represents zero induced unbalance within the system as no weight was installed on the unbalance disc, while 4 represents the maximum unbalance within the system where 6.614 grammes of weight were installed at the recess that is located 23 mm from the centre of the unbalance disc. Each intensity was divided into two classes, denoted by the letters

D and E, where the former represents the 'Development' dataset and the latter, the 'Evaluation' dataset, respectively. The development dataset is to be used for training ML models, while the evaluation datasets are to be used by trained black boxes to perform fault prediction. Each of them was recorded for a specific amount of time, with each sample being recorded in one second with 4,096 sampling rate. The 'Number of samples' values represent both the total number of samples and the duration of the datasets recorded. 6434 of a 1D dataset, for example, denotes that the data was recorded for 6,434 seconds, or approximately 1 hour and 47 minutes of recording time, and comprises 26,353,664 data values.

Table 2. Unbalance Datasets for Testing by Mey et al. [15].

Identity	Radius (mm)	Mass (g)	Unbalance factor (mm.g)	Number of samples	
				Evaluation (E) (testing)	Total number of data
0E	-	0	0	1,670	6,840,320
1E	14 ± 0.1	3.281 ± 0.003	45.9 ± 1.4	1,673	6,852,608
2E	18.5 ± 0.1	3.281 ± 0.003	60.7 ± 1.9	1,669	6,836,224
3E	23 ± 0.1	3.281 ± 0.003	75.5 ± 2.3	1,672	6,848,512
4E	23 ± 0.1	6.614 ± 0.007	152.1 ± 2.3	1,675	6,860,800

From the provided data, two parameters, Crest Factor and Kurtosis were extracted to be the training materials for the RNN which were synthesised by using the following equations:

$$RMS = \sqrt{\left(\frac{1}{N}\sum_{i=1}^{N}(x(i) - \bar{x})^2\right)} \tag{1}$$

$$Peak\ to\ valley = \max\big(x(n)\big) - \min\,(x(n)) \tag{2}$$

$$Crest\ Factor = \frac{Peak\ to\ valley}{RMS} \tag{3}$$

$$Kurtosis = \frac{\frac{1}{N}\sum_{i=1}^{N}(x(i) - \bar{x})^4}{RMS^4} \tag{4}$$

Note: N = length of the signal; and $\overline{x}$ = mean value of the data.

Because the RNNs in this study were built using MATLAB R2017B on a Windows 64-bit i5 CPU with 8GB RAM, all of the extracted parameters from the above equations must be sorted out in the matrix array in order for MATLAB to compute them as shown below:

$$\begin{bmatrix} CF1_1 & CF2_1 & CF3_1 & K1_1 & K2_1 & K3_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ CF1_{6430} & CF2_{6430} & CF3_{6430} & K1_{6430} & K2_{6430} & K3_{6430} \end{bmatrix} \tag{5}$$

Note that,

CF1 = Crest Factor of vibration data from sensor 1 (DC motor)

CF2 = Crest Factor of vibration data from sensor 2 (bearing pedestal vertical)

CF3 = Crest Factor of vibration data from sensor 3 (bearing pedestal horizontal)

K1 = Kurtosis of vibration data from sensor 1 (DC motor)

K2 = Kurtosis of vibration data from sensor 2 (bearing pedestal vertical)

K3 = Kurtosis of vibration data from sensor 3 (bearing pedestal horizontal)

And target matrix for RNN is made of the values of unbalance factor (UF) from Table 1. Because Mey et al. data contained a variable total number of samples, the authors chose 6430 for training purposes in order to ensure that all matrices had a consistent total number of rows, while 1670 was chosen for testing purposes. All constructed RNNs were trained on development (D) datasets and tested on evaluation (E) datasets.

$$\begin{bmatrix} UF_1 \\ \vdots \\ UF_{6430} \end{bmatrix} \tag{6}$$

RNN Architectures

As per the recommendation by MathWorks® in its documentation, it is suggested that the construction of RNN begins with an open loop architecture (Figure 2 (top)) for training purposes and later be converted into a close loop (Figure 2 (bottom)) for multistep-ahead prediction [25]. Although the recommendation was to begin with an open loop, this scenario triggered a new hypothesis regarding the prediction outcome when a closed loop comes into consideration for the training stage. This study also considered the hypothesis that retraining the black box would improve the predicting performance, making it more accurate than the black boxes that only undergo one training session.

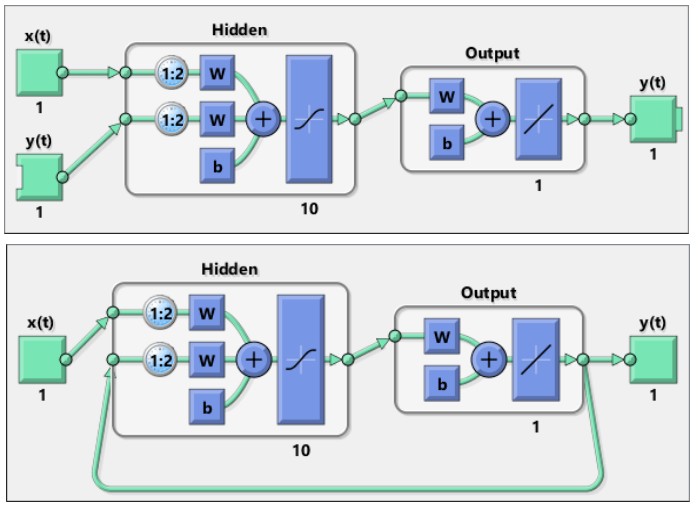


Figure 1: Open Loop RNN (top), close loop RNN (bottom) – generated in MATLAB.

Prediction Accuracy

Each prediction from the black box will be taken into comparison against the designated unbalance factor and the tolerances of -0.1 and +0.1 for maximum and minimum were given for any deviated values to be considered as correct, respectively. For example, if the unbalance factor for 1E was 45.9 mm.g and the prediction came out with a value of 45.8 mm.g or 46.0 mmg, the value will be considered correct, but if the value goes less than or beyond the tolerance given, the prediction will not be taken into account as correct. Overall accuracy is calculated by applying the following equation:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{7}$$

RNN CLASSIFICATIONS

Approach 1: RNN Through Bulk Datasets

There were several parametric selections to be made in the construction of an RNN that fulfilled the aim of predicting unbalanced fault accurately. Parameters such as input delays, type of training function, number of hidden layers, quantity of neurons within every layer, type of learning function, type of performance function, and number of retraining sessions for the black box need to be conducted. The impact of these parameters was studied through multiple iterations and was conducted through mix and match and trial and error to obtain the best prediction analysis. Furthermore, the way datasets are created also plays a major role in producing effective training sessions to produce black boxes with the prediction capability of matching the target matrix. In this approach, all extracted parameters from the development (D) group of every unbalanced intensity were stacked altogether to produce a very long matrix array of inputs, as shown below. A similar observation was seen for the evaluation (E) stage. The matrix array of evaluation (E) was made in the same manner as development (D), where they were also stacked altogether, but only for input because the values of output would be monitored afterwards to evaluate its accuracy.

Approach 1.1: Open-loop vs close-loop

This approach was the expansion of Approach 1, where it was carried out by creating a few architectures that each were developed into two classes; open loop and close loop architectures. The prior architecture will be created for the training stage and will be converted into a close loop for fault prediction, while the latter architecture will be created without being converted for fault prediction. The prediction prowess will be put into comparison with each other subsequently.

Approach 1.2: Retraining the black box

This approach was also a branch of what was produced within Approach 1 and was carried out by creating a few architectures that will undergo the training process five times, which means the produced black box from the first training session will be taken into another four training sessions before the evaluation (E) arrays is fed into them for fault prediction purposes.

$$\underbrace{\begin{bmatrix} CF1_{0D_1} & CF2_{0D_1} & CF3_{0D_1} & K1_{0D_1} & K2_{0D_1} & K3_{0D_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ CF1_{0D_{6430}} & CF2_{0D_{6430}} & CF3_{0D_{6430}} & K1_{0D_{6430}} & K2_{0D_{6430}} & K3_{0D_{6430}} \\ CF1_{1D_1} & CF2_{1D_1} & CF3_{1D_1} & K1_{1D_1} & K2_{1D_1} & K3_{1D_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ CF1_{1D_{6430}} & CF2_{1D_{6430}} & CF3_{1D_{6430}} & K1_{1D_{6430}} & K2_{1D_{6430}} & K3_{1D_{6430}} \\ CF1_{2D_1} & CF2_{2D} & CF3_{2D_1} & K1_{2D_1} & K2_{2D_1} & K3_{2D_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ CF1_{2D_{6430}} & CF2_{2D_{6430}} & CF3_{2D_{6430}} & K1_{2D_{6430}} & K2_{2D_{6430}} & K3_{2D_{6430}} \\ CF1_{3D_1} & CF2_{3D_1} & CF3_{3D_1} & K1_{3D_1} & K2_{3D_1} & K3_{3D_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ CF1_{3D_{6430}} & CF2_{3D_{6430}} & CF3_{3D_{6430}} & K1_{3D_{6430}} & K2_{3D_{6430}} & K3_{3D_{6430}} \\ CF1_{4D_1} & CF2_{4D_1} & CF3_{4D_1} & K1_{4D_1} & K2_{4D_1} & K3_{4D_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ CF1_{4D_{6430}} & CF2_{4D_{6430}} & CF3_{4D_{4096}} & K1_{4D_{6430}} & K2_{4D_{6430}} & K3_{4D_{6430}} \end{bmatrix}}_{\text{Input}} \quad \underbrace{\begin{bmatrix} UF_{0D_1} \\ \vdots \\ UF_{0D_{6430}} \\ UF_{1D_1} \\ \vdots \\ UF_{1D_{6430}} \\ UF_{2D_1} \\ \vdots \\ UF_{2D_{6430}} \\ UF_{3D_1} \\ \vdots \\ UF_{3D_{6430}} \\ UF_{4D_1} \\ \vdots \\ UF_{4D_{6430}} \end{bmatrix}}_{\text{Target}} \tag{8}$$

Approach 2: RNN Through Separate Datasets

As illustrated in Figure 3, this stage can only begin after obtaining the results from Approach 1, which confirms the effects of parametric studies from varying the listed methods (training function, amount of input delays and feedback delays, number of hidden layers, neuron quantities, learn function, and retrain amount) on prediction prowess. The best parameters that produced the best predictions amongst all will be taken forward into creating new RNNs within this approach. The new RNN will be trained only by using one class of unbalanced development (D) datasets per training session and will perform the prediction on the respective evaluation (E) datasets with the same unbalanced intensity.

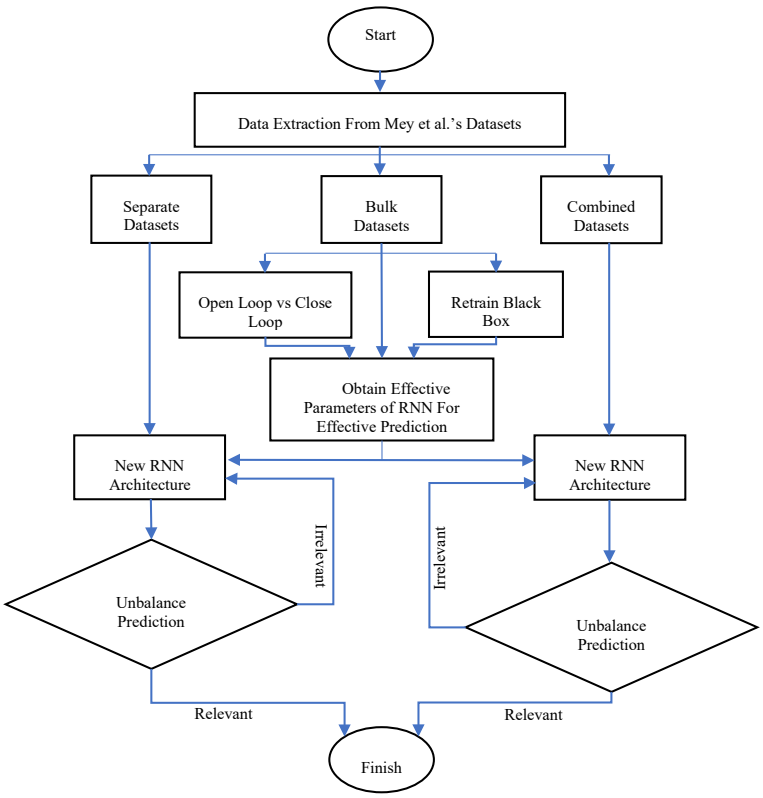


Figure 3. Flow process of RNN for unbalance prediction.

Another RNN with the same architecture will be created anew and will be trained using the next class of unbalance intensity from development (D) datasets and will perform the prediction on the evaluation (E) datasets from the next class of unbalance intensity as well. This process was repeated until maximum unbalance intensity is reached as per depicted in Table 3 and finished by taking the prediction results into comparison against the designated unbalance factor, as shown in Table 1.

Table 3. Architecture for RNN via bulk datasets.

Bill	Methods (loop)		Training function	Delays		Hidden layers	Neuron allocations		Learn function	Retrain (no. of retrain)
	Training	Prediction		Input	Feedback		Layer	Quantity		
1	Open	Close	Trainlm	2:9	3:7	3	1	29	Learngd	No
							2	29	Learngdm	
							3	23	Learngd	
2	Open	Close	Trainlm	0:37	0:67	3	1	29	Learngd	No
							2	29	Learngdm	
							3	23	Learngd	
3	Open	Close	Trainlm	0:9	1:11	4	1	7	Learngd	No
							2	5	Learngdm	
							3	3	Learngd	
							4	11	Learngdm	
4	Open	Close	Trainbr	0:9	1:11	2	1	10	Learngdm	No
							2	10	Learngdm	
5	Open	Close	Trainrp	0:9	1:11	2	1	29	Learngdm	Yes (4)
							2	37	Learngdm	
6	Open	Close	Traincgf	0:9	1:11	2	1	29	Learngdm	Yes(4)
							2	37	Learngdm	
7	Open	Close	Traingd	0:9	1:11	2	1	29	Learngdm	Yes (4)
							2	37	Learngdm	
8	Open	Close	Traingdm	0:9	1:11	2	1	29	Learngdm	Yes (4)
							2	37	Learngdm	
9	Close	Close	Trainlm	0:1	1:2	2	1	13	Learngdm	Yes (1)
							2	17	Learngdm	
10	Close	Close	Trainlm	0:1	1:2	2	1	31	Learngdm	Yes (1)
							2	11	Learngdm	
11	Close	Close	Trainlm	2:3	1:4	2	1	3	Learngdm	No
							2	7	Learngdm	
12	Close	Close	Trainlm	4:5	1:5	2	1	3	Learngdm	No
								7	Learngdm	

Approach 3: RNN Through Combined Datasets

Table 4 shows how this combined datasets strategy is crafted. It is a replication of what Oliver Mey and his colleagues did to train their ML models since this work benchmarked the datasets and a few ML algorithms from Mey et al. The datasets were created by mixing the zero unbalance intensity dataset with other datasets from various levels of unbalance intensity. For training, a 0D dataset was coupled with a 1D dataset, a 0D dataset with 2D, a 0D dataset with 3D, and a 0D dataset with 4D. The same is true for testing. The dataset was produced by coupling 0E with 1E, 0E with 2E, 0E with 3E, and 0E with 4E.

Table 4. Architectures for RNN through separate datasets and combined datasets.

Bill	Methods (loop)		Training function	Delays		Hidden layers	Neuron allocations		Learn function	Retrain (no. of retrain)
	Training	Prediction		Input	Feedback		Layer	Quantity		
1	Close	Close	Trainlm	1:5	3:5	3	1	3	Learngdm	No
							2	7	Learngdm	
2	Close	Close	Trainlm	1:5	3:5	3	1	3	Learngdm	No
							2	7	Learngdm	
							3	5	Learngdm	
3	Close	Close	Trainlm	1:5	3:5	4	1	3	Learngdm	No
							2	7	Learngdm	
							3	5	Learngdm	
							4	11	Learngdm	

Note that,
Trainlm = Lavenberg-Marquardt Train Function,

Trainbr = Bayesian Regularisation Train Function,
Traingd = Gradient Descent Train Function,
Traingdm = Gradient Descent with Momentum Train Function,
Trainrp = Resilient Backpropagation Train Function
Traincgf = Fletcher-Powell Conjugate Gradient
Learngd = Gradient Descent Learn Function
Learngdm = Gradient Descent with Momentum Learn Function
All architectures for approach 1, 2 and 3 within this study utilised mean square error (MSE) as a performance function.

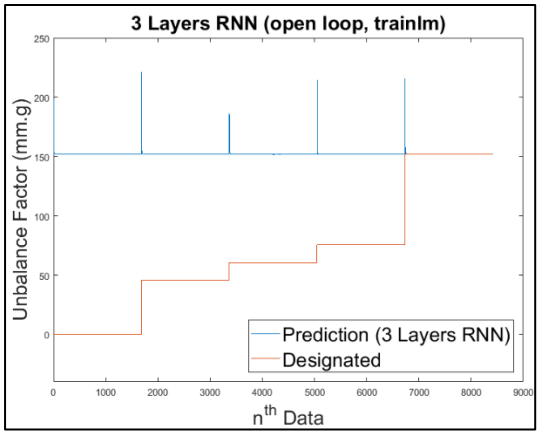
RESULTS AND DISCUSSION

In this section, the predictions from a variety of RNN classifications were discussed sequentially, as per the previous section.

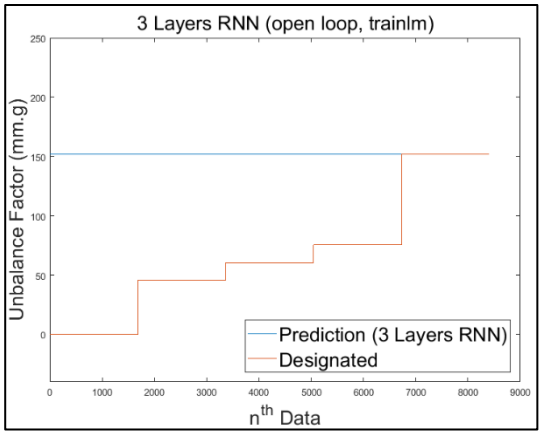
RNN Through Bulk Datasets

Figure 4 and 5 show the results of fault prediction training using bulk datasets. The graphs from (A1) to (A12) show the predictions made by RNNs were trained by using bulk datasets, as per shown in Eq. (8). Each prediction has an accuracy of 20.03%, 20.03%, 20.03%, 1.07%, 5.31%, 7.88%, 19.65%, 19.84%, 0.58%, 0.15%, 1.22%, and 0.07%, respectively, for A1 to A12. Within the group of open-loop and close-loop designs, black boxes created with the Lavenberg-Marquardt train function (trainlm) produced the best forecasts.

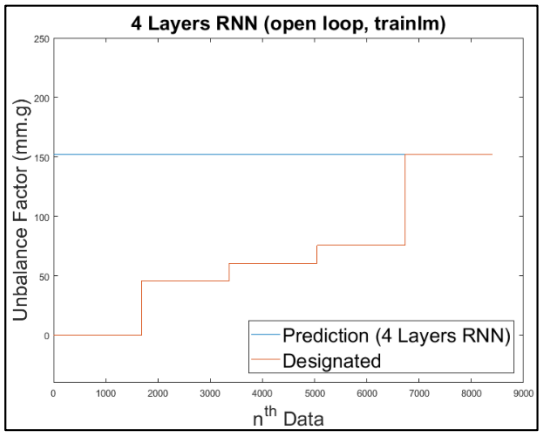
Furthermore, because the trainlm produced the best predictions for open loop and close loop, with accuracy of 20.03% and 1.22%, respectively, the predictions for open loop tended to stay in the region of 150 mm.g regardless of unbalance intensity, whereas the predictions for close loop responded more significantly to the increment of unbalance factor. Although the close loop appears to have insignificant accuracy, the authors believe that the close loop architecture will be significant to the unbalance fault prediction in the separate datasets and combined datasets approaches because of its behaviour in producing predictions that mimic the designated unbalance factor more dynamically than open loop architectures. In terms of delay, their effects may be seen in the results in A1, A2, and A3, where different values of delays had no effect on prediction results, according to A1, A2, and A3, rather than slowing down the processing time. Consequently, the incoming approaches from B and C will be carried out with shorter delays to prevent time loss.



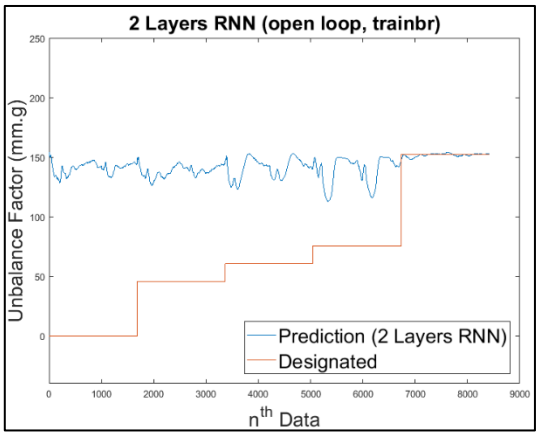
(A1)



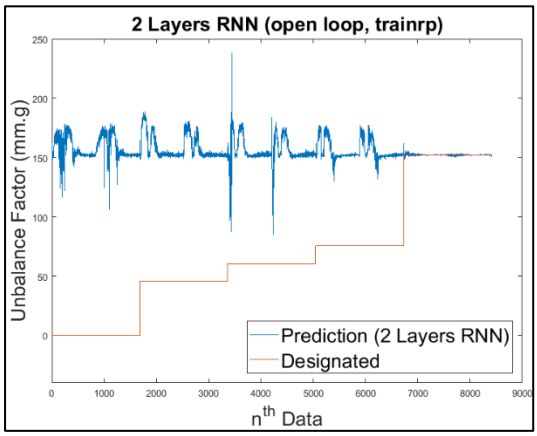
(A2)



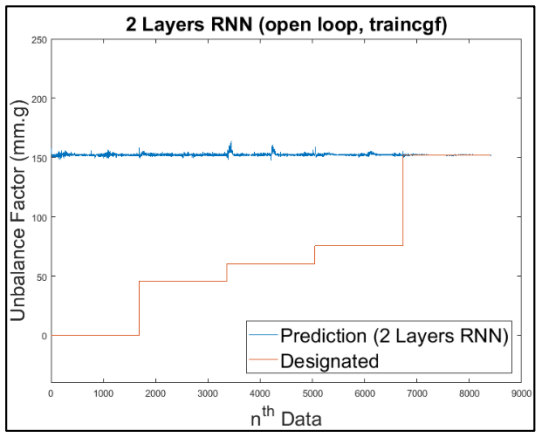
(A3)



(A4)



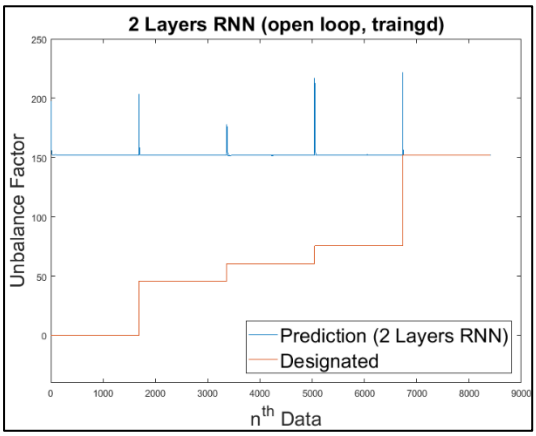
(A5)



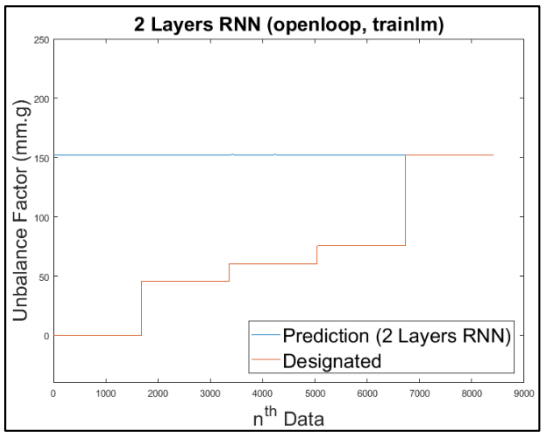
(A6)

Figure 4. The above figures are sequential representations of the RNN architecture shown in Table 2, where (A1), (A2), (A3), (A4), (A5), and (A6) represent architectures A1, A2, A3, A4, A5, and A6, respectively. The graphs show how much the predictions (blue lines) made by RNN deviate from the designated value (orange lines).

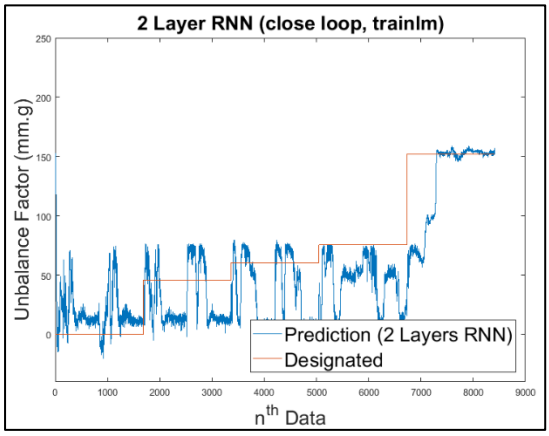
The number of neurons in each layer was taken into account as an early hypothesis, with the expectation that the number of neurons created would be proportional to the prediction outcome, with the higher the number of neurons produced, the better the prediction accuracy. However, the open-loop pool's A1, A2, A3, A6, and A7 indicated that topologies with a higher number of neurons per layer did not provide better predictions than those with a lower number of neurons per layer. When it comes to the effect of retraining on fault prediction performance, RNNs with architectures of A5, A6, A7, A8, A9, and A10 show no significant improvement.



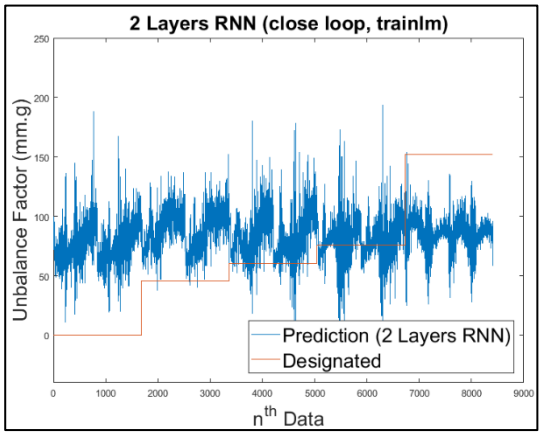
(A7)



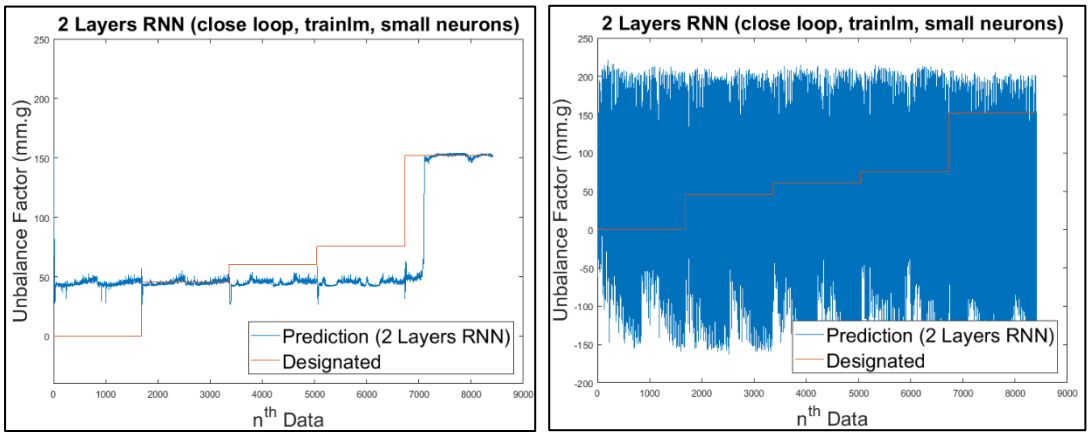
(A8)



(A9)



(A10)



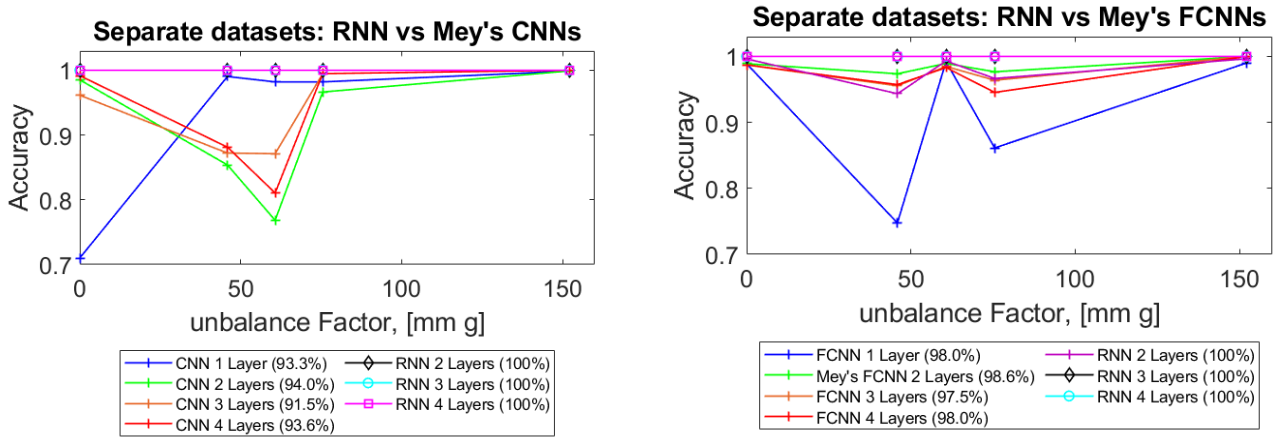
(A11) (A12)

Figure 5. The above figures are sequential representations of the RNN architecture shown in Table 2, where (A7), (A8), (A9), (A10), (A11), and (A12) represent architectures A7, A8, A9, A10, A11, and A12, respectively. The graphs show how much the predictions (blue lines) made by RNN deviate from the designated value (orange lines).

Although there were some minor variations in the prediction patterns, as observed in A5 of the open-loop and A9 of the close loop, the predictions in A10 became unpredictable and had a forecast accuracy of less than 1%. As a result, retraining failed to improve the overall accuracy of the selected black boxes. The twelfth (A12) RNN architecture's results, which have gone beyond A10's erratic, show the obvious alterations in the unbalance prediction. This architecture can be compared to the eleventh (A11) RNN architecture because they both used the same train function, had the same number of layers, and had the same number of neurons in each layer, but they differed in terms of input delay allocation, with A12 receiving four (4) delayed inputs and A11 receiving two (2) delayed inputs. The amount of time it takes for the output to respond to the input is referred to as input delays. As a result, these two results show how the amount of delays for the close loop RNN architecture affects fault prediction accuracy, with longer input delays reducing fault prediction accuracy more than shorter input delays.

RNN Through Separate Datasets

The strategy of using datasets in this approach was similar to that utilised by Oliver Mey and his colleagues, who trained CNNs and FCNNs using datasets of different unbalance intensities. In order to obtain a clear contrast in terms of prediction ability, the RNN prediction results were overlayed together with those produced by Mey et al. Figure 6 shows that RNNs trained on independent datasets were able to accurately predict the unbalance and achieve an overall accuracy of 100%, or 1.00 in efficiency, outperforming CNNs and FCNNs in every unbalance intensity. Although both graphs B1 and B2 show that both CNNs and FCNNs achieved convergence at the highest unbalance intensity, the low prediction accuracies at unbalance intensities of 0, 1, 2, and 3 have reduced the overall unbalance prediction percentage, with CNNs having 93.3% (1 layer), 94.0% (2 layers), 91.5% (3 layers), 93.6% (4 layers), and FCNNs having 91.6% (0 layer), 98.0% (1 layer), 98.6% (2 layers), 97.5 (4 layers).



(B1) (B2)

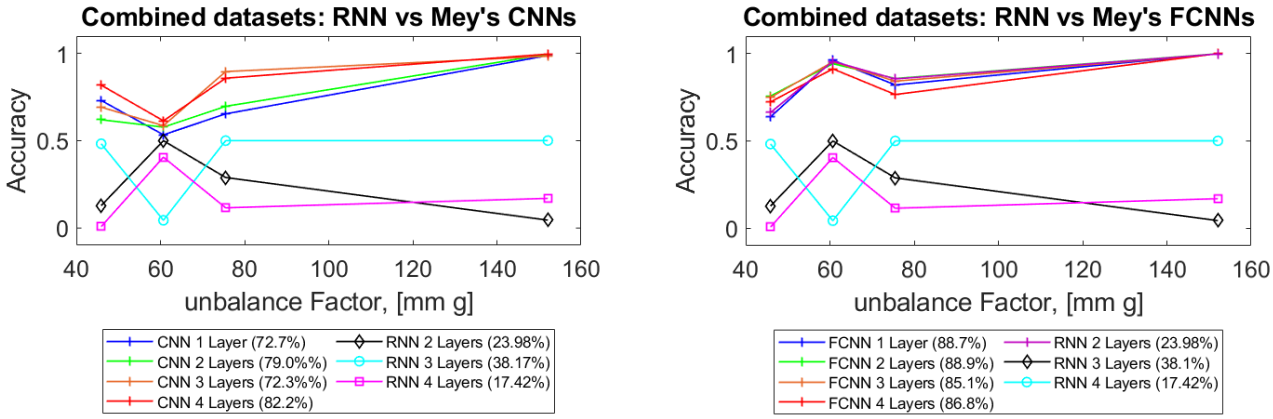
Figure 6. RNNs' fault prediction against convolutional neural networks (B1), fully-connected neural networks (B2) through separate datasets approach.

Regardless of the number of layers and neurons inside each layer, the RNN architectures presented in Table 3 are capable of making correct predictions. Neuron amounts were tabulated at random in each layer in this technique, while

the rest of the parameters were set to fixed values. This method has yielded black boxes capable of reliably predicting the unbalance fault, with the datasets specifically prepared based on the unbalance intensity having a substantial impact on unbalance prediction. As such, the method of preparation of datasets has become a vital factor for consideration in the development of effective RNN models.

RNN Through Combined Datasets

The findings of this strategy were also overlayed with what Mey et al. achieved in training their CNNs and FCNNs using combined datasets of 0E with 1E, 0E with 2E, 0E with 3E, and 0E with 4E, similar to Approach 2. On every combined dataset, RNNs scored 23.98% (2 layers), 38.17% (3 layers), and 17.42% (4 layers), while CNNs had an accuracy of 72.7% (1 layer), 79.0% (2 layers), 72.3% (3 layers), and 82.2% (4 layers), and FCNNs had an accuracy of 85.6% (0 layer), 88.7% (1 layer), 88.9% (2 layers), 85.1% (3 layers), and 86.8% (4 layers). When comparing the fault predictions of Approach 3 to those of Approach 1 and 2, it was found that the fault predictions of Approach 3 were better than those created by RNNs using bulk datasets but much worse than those produced using separate datasets. In terms of fault prediction accuracy when using combined datasets, accuracy graphs reveal that Mey et al.'s CNNs and FCNNs also suffer from deterioration albeit not to the same extent as the RNNs. According to Figure 7, the order in which datasets were stacked has a significant impact on the accuracy of RNN's unbalance prediction, as RNN appears to have a difficult time correctly predicting the output when multiple datasets from different unbalance intensities are stacked together, as done in Approach 1.



(C1) (C2)

Figure 7. RNNs' fault prediction against convolutional neural networks (B1), and fully-connected neural networks (B2) through combined datasets approach.

CONCLUSION

In this research, a number of RNN designs for unbalance fault prediction are proposed, each of which takes into account different unbalance intensities in the published data. There are three basic approaches to the proposed method: To begin, the models were created using bulk datasets, in which each dataset from different unbalance intensities were stacked together in the form of a long matrix array to be fed into the RNN, which was then split into two minor approaches: considering open-loop vs close-loop parameters and retraining the created black boxes. Second, based on the best architecture and parameters from the previous approach, new RNN models were created and trained using datasets that were divided into groups according to unbalance intensity. Finally, RNN models were created by utilising combined datasets from different unbalance intensities.

This paper's strategy focuses on RNN architectures that were developed in a random yet structured manner while taking into account several parameters to create black boxes capable of providing high prediction accuracy under varied dataset designs. Multiple parameters, such as the type of loop, training function type, amount of delays, neuron quantity, and the way datasets are prepared, must be taken into account, according to the analysis, because all of these have contributed to accurate black box predictions where RNNs that used separate datasets were able to perform accurate unbalance predictions whenever datasets that were segregated into the group of unbalance intensity were provided. The longer the input delays were employed, the worse the prediction, whereas the number of layers and neuron quantity had no effect on the prediction. The Lavernberg-Marquardt training function and the Gradient Descent with Momentum learning function, out of all the training and learning functions employed, contributed to the creation of black boxes with great prediction accuracy. Finally, retraining the accomplished black boxes would not increase their prediction accuracy considerably.

This work did not use the end-to-end technique without feature extraction used by Chen et al. and Jiang et al., in which the prior was performed using Cyclic Spectral Coherence (CSCo), which converts raw data into a 2D map, which is subsequently processed by CNN utilising pattern recognition [26]. The latter carried out their fault diagnosis research by feeding the inputs obtained by converting raw vibration data into vibration spectrum using the Fast Fourier Transform (FFT) [10]. Because the goal of this study was to create an RNN capable of recognising unbalance with the least number

of statistical moments possible (just two statistical moments were used throughout the study), such an automated conversion was not used. Furthermore, to the author's knowledge, there has never been a mechanism for automatically extracting statistical moments for use as machine learning input. This study uses the manual extraction approach to derive the crest factor and kurtosis.

In summary, RNN can be the best alternative for reliable unbalance fault prediction if the datasets are separated into different datasets using unbalanced intensity clustering. In terms of dealing with sequential data, RNN outperforms CNN and FCNN since the findings demonstrate 100% prediction accuracy. The positive findings of this study will be carried forward into the future by using phase shift as a new addition to the existing parameters in order to increase prediction capabilities using the bulk datasets method.

ACKNOWLEDGEMENT

The authors would like to thank Universiti Malaysia Pahang (www.ump.edu.my) for providing financial support and laboratory facilities under Post-Graduate Research Scheme (PGRS) No. PGRS210383 and Research Grant Scheme (RDU) No. RDU210335.

REFERENCES

[1] D. Goyal and B. Pabla, "Condition based maintenance of machine tools - A review, " *CIRP J. Manuf. Sci. Technol .,* vol. 10, pp. 24-35, 2015, doi: 10.1016/j.cirpj.2015.05.004.

[2] J. Gold, "How the oil and gas industry exploits IoT," Network World, 11 October 2019. [Online]. Available: https://www.networkworld.com/article/3445204/how-the-oil-and-gas-industry-exploits-iot.html. [Accessed: Oct 29, 2021].

[3] A. Alugboji *et al.*, "Development of a low-cost smart pig and wireless sensor for the detection of pipeline defects and anomalies," *ABUAD J. Eng. Res. Dev. (AJERD),* vol. 3, no. 1, pp. 68-75, 2020.

[4] E. Aba *et al.*, "Petroleum pipeline monitoring using an internet of things (IoT) platform," *SN Appl. Sci.,* vol. 3, pp. 180, 2021, https://doi.org/10.1007/s42452-021-04225-z.

[5] M. Mohammadpoor and F. Torabi, "Big data analytics in oil and gas industry: An emerging trend," *Petroleum,* vol. 6, no. 4, pp. 321-328, 2020, doi: 10.1016/j.petlm.2018.11.001.

[6] H. Xu, R. Ma, L. Yan and Z. Ma, "Two-stage prediction of machinery fault trend based on deep learning for time series analysis," *Digit. Signal Process.,* vol. 117, p. 103150, October 2021, doi: 10.1016/j.dsp.2021.103150.

[7] T. Hasegawa, M. Saeki, T. Ogawa and T. Nakano, "Vibration-based fault detection for flywheen condition monitoring," in *ICSI 2019 The 3rd International Conference on Structural Integrity*, Madeira, 2019.

[8] Case Western Reserve University, "Bearing data center," Case Western Reserve University, [Online]. Available: https://engineering.case.edu/bearingdatacenter/welcome. [Accessed: May 16, 2021].

[9] B. Zhao, X. Zhang, H. Li and Z. Yang, "Intelligent fault diagnosis of rolling bearings based on normalised CNN considering data imbalance and variable working conditions," *Knowl. Based Syst.,* vol. 199, pp. 105971, 2020, doi: 10.1016/j.knosys.2020.105971.

[10] H. Jiang, X. Li, H. Shao and K. Zhao, "Intelligent fault diagnosis of rolling bearing using improved deep recurrent neural network," *Meas. Sci. Technol.,* vol. 29, no. 6, pp. 065107, 2018, doi: 10.1088/1361-6501/aab945.

[11] Y. Xie and T. Zhang, "A long short term memory recurrent neural network approach for rotating machinery fault prognosis," in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, Xiamen, 2018.

[12] J. Ben Ali *et al.*, "Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals," *Appl. Acoust.,* vol. 89, pp. 16-27, 2015, doi: 10.1016/j.apacoust.2014.08.016.

[13] P. Lecinski, "Bearing problems - fault frequency and artificial intelligence-based methods," CBM Connect, 18 October 2021. [Online]. Available: https://www.cbmconnect.com/bearing-problems-fault-frequency-and-artificial-intelligence-based-methods/. [Accessed: Oct. 28, 2021].

[14] O. Mey, W. Neudeck, A. Schneider and O. Enge-Rosenblatt, "Vibration measurements on a rotating shaft at different unbalance strengths," Fraunhofer, 25 March 2020. [Online]. Available: https://fordatis.fraunhofer.de/handle/fordatis/151.2. [Accessed: April 15, 2021].

[15] O. Mey, W. Neudeck, A. Schneider and O. Enge-Rosenblatt, "Machine learning-based unbalance detection of a rotating shaft using vibration data," in *25th International Conference on Emerging Technologies and Factory Automation*, Vienna, 2020.

[16] S. Sharma, W. Abed, R. Sutton and B. Subudhi, "Corrosion fault diagnosis of rolling element bearing under constant and variable load and speed conditions," in *9th IFAC Symposium on Control of Power and Energy Systems CPES 2015*, New Delhi, 2015.

[17] Y. Zhang, L. He, J. Yang, J. Wang and G. Zhu, "Vibration control of tie rod rotors with optimisation of unbalanced force and unbalance moment," *IEEE Access,* vol. 8, pp. 66578-66587, 2020.

[18] D. Huang, "Characteristics of torsional vibrations of a shaft with unbalance," *J. Sound Vib.,* vol. 308, no. 3-5, pp. 692-698, 2007, doi: 10.1016/j.jsv.2007.04.005.

[19] Y. Meng, L. Lu and J. Yan, "Shaft orbit feature based rotator early unbalance fault identification," in *9th International Conference on Digital Enterprise Technology - DET 2016 - "Intelligent Manufacturing in the Knowledge Economy Era"*, Nanjing, 2016.

[20] P. Gangsar and R. Tiwari, "A support vector machine based fault diagnostics of Induction motors for practical situation of multi-sensor limited data case," *Measurement,* vol. 135, pp. 694-711, 2019, doi: 10.1016/j.measurement.2018.12.011

[21] J. Yan, Y. Hu and C. Guo, "Rotor unbalance fault diagnosis using DBN based on multi-source heterogeneous information fusion," in *2nd International Conference on Sustainable Materials Processing and Manufacturing (SMPM 2019)*, Sun City, 2019. .

[22] B. Liu, W. Zhang, X. Xu and D. Chen, "Time delay recurrent neural network for speech recognition," in *2019 3rd International Conference on Machine Vision and Information Technology (CMVIT 2019)*, Guangzhou, 2019.

[23] N. K. Vitanov, N. P. Hoffman and B. Wernitz, "Nonlinear time series analysis of vibration data from a friction brake: SSA, PCA, and MFDFA," *Chaos, Solitons & Fractals: Nonlinear Science, and Nonequilibrium and Complex Phenomena,* vol. 69, pp. 90-99, 2014, doi: 10.1016/j.chaos.2014.09.010.

[24] T. Navamani, "Efficient deep learning approaches for health informatics," in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, Missouri, Academic Press, 2019, pp. 123-137.

[25] The MathWorks, Inc., "narxnet," The MathWorks, Inc., [Online]. Available: https://www.mathworks.com/help/deeplearning /ref/narxnet.html. [Accessed: July 21, 2021].

[26] Z. Chen, A. Mauricio, W. Li and K. Gryllias, "A deep learning method for bearing fault diagnosis based on cyclic spectral coherence and convolutional neural networks," *Mech. Syst. Signal Process.,* vol. 140, p. 106683, 2020, doi: 10.1016/j.ymssp.2020.106683.

[27] C. Wu, P. Jiang, C. Ding, F. Feng and T. Chen, "Intelligent fault diagnosis of rotating machinery based on one-dimensional convolutional neural network," *Comput Ind.,* vol. 108, pp. 53-61, 2019, doi: 10.1016/j.compind.2018.12.001.

[28] F. L. Santos, M. L. Machado Duarte, M. T. C. de Faria and A. C. Eduardo, "Balancing of a rigid rotor using artificial neural network to predict the correction masses," *Acta Sci. Technol.,* vol. 31, no. 2, pp. 151-157, 2009, doi: 10.4025/actascitechnol.v31i2.3912.

[29] W. Wu and X. Lu, "Retrieving information and discovering knowledge from unstructured data using big data mining technique: heavy oil fields example," in *International Petroleum Technology Conference*, Kuala Lumpur, 2014.

[30] J. Feblowitz, "Analytics in oil and gas: the big deal about big data," in *2013 SPE Digital Energy Conferences and Exhibition*, Texas, 2013.

[31] J. Liu, R. Hao, T. Zhang and X. Wang, "Vibration fault diagnosis based on stochastic configuration neural networks," *Neurocomputing,* vol. 434, pp. 98-125, 2021, doi: 10.1016/j.neucom.2020.12.080.

[32] H. Liu *et al.*, "Fault diagnosis of rolling bearings with recurrent neural network-based autoencoders," *ISA Transactions,* vol. 77, pp. 167-178, 2018, doi: 10.1016/j.isatra.2018.04.005.

[33] B. Crockett and K. Kurrey, "Smart decision making needs automated analysis: Making sense out of big data in real-time," in *SPE Intelligent Energy Conference and Exhibition*, Utrecht, 2014.