Sensors and Materials, Vol. 36, No. 4 (2024) 1419–1430
MYU Tokyo

S & M 3608

A Small Deep Learning Model for Fault Detection of a Broken Rotor Bar of an Induction Motor

Pat Taweewat, Warachart Suwan-ngam,
Kanoknuch Songsuwankit, and Poom Konghuayrob*

School of Engineering, King Mongkut's Institute of Technology Ladkrabang
1, Soi Chalongkrung 1, Ladkrabang, Bangkok 10520, Thailand

(Received January 8, 2024; accepted April 2, 2024)

Keywords: broken rotor bar detection, MCSA, FFT feature extraction, deep learning, model reductions

In this paper, we present an investigation of a small deep learning model applied to the detection of a broken rotor bar of an induction motor. The motor current spectrum analysis is the base method for fault detection. This proposed method focuses on the analysis of the modification of the input vector and model configuration. This method was implemented and it showed that the feature length and size of the model are reduced compared with the existing method. The experimental results showed that only feature extraction using the spectral-based method and limit range of its coefficient are adequate to provide accuracy of small deep learning comparable to that of the parallel-layer deep learning model. Likewise, at the same accuracy level, based on the deep learning model, a shorter sampling duration than that required by the reference model is needed.

1. Introduction

Induction motors are widely used in industry because they are versatile and easy to control and have few maintenance parts. Faults in these motors can lead to various catastrophes such as income loss, fatal accidents, and high maintenance cost. Early detection of these faults may prevent these catastrophes by enabling the scheduling of maintenance in advance. Electrical signals in either the time domain or frequency domain are applied to detect the fault,[1] for example, vibration source,[2] current,[3] heat generated by the motor,[4] and Park's vectors.[5] Because various types of fault can occur, investigations usually focus on particular types of fault.[6-8] The broken rotor bar (BRB) fault is one of the main faults leading to motor failure, and it is considered to make up around 5–10% of all the faults described in the literature.[3,6,9] Although fault detection using vibration signals produces a good result, the restriction on the location at which to install an accelerometer can be a problem.[2] Operation downtime to install the sensor is also required. An alternative method is to apply the motor current signature analysis (MCSA). This method requires only the stator current and is one of the most popular methods because it is convenient to run the analysis online without disconnecting the electrical circuit.[10]

*Corresponding author: e-mail: poom.ko@kmitl.ac.th
https://doi.org/10.18494/SAM4847

ISSN 0914-4935

However, the shortcoming of using the MCSA is the difficulty in identifying the right peak associated with fault types, and experts in the field are required to perform the MCSA fault diagnosis. Using machine learning models to replace these experts is useful as this method causes maintenance departments in industry to perform preliminary fault detection while waiting for the full diagnosis by an expert. In previous studies, many machine learning techniques were shown to have good detection results.[2,3,8,11] Such techniques include the deep learning method,[2,3,8] which is a method used in several applications because of its ability to improve its performance over a large dataset.[12,13]

The deep learning for BRB fault detection based on the current spectrum and signal envelope features did show good detection results, as reported in a previous study.[3] However, its drawback is clear as the model size is quite large for the problem, and a long duration is required for each current sampling. Consequently, the long detection time is required to yield a fault alarm. As suggested in a review study,[6] the early state of the fault should be detected; hence, the signal duration used for fault detection should be as short as possible. Thus, this method needs to be optimized to resolve these issues. In previous studies,[2-3,8] the FFT spectral feature reduction and effect of different deep learning architectures, which are the main points we investigate in this research, were not examined. In our study, we propose an improved method to decrease the sampling duration and model size. Experiments will be conducted to verify the results.

To provide an overview of the previous method and the proposed modification, this paper is organized as follows: the deep learning fault detection is explained in Sect. 2 and the proposed method is discussed in Sect. 3. The experiment to validate the method is presented in Sect. 4 and the results and discussion are given in Sect. 5. Finally, the conclusion is presented in Sect. 6.

2. Deep Learning Fault Detection

Deep learning has been used for several types of fault detection, including BRB fault detection.[3] In this section, we mainly focus on deep learning as well as the elements required for fault detection.

2.1 Machine learning and deep learning

Machine learning is a method used to perform tasks such as classification and prediction using a model generated by a data-driven method.[12] The common elements found in machine learning are feature extraction and the machine learning model. Since the good performance of the model depends on many factors, including the input vector generated from feature extraction, some effort is required to find a suitable feature extraction for one specific problem. In contrast, deep learning is a technique applied in machine learning, wherein multiple layers of a neural network are used to produce learning features for the final classifier.[13,14]

Thus, the feature-learning ability of deep learning is used to find the suitable feature through data rather than analysis by a human, but a large dataset is required.[15] For implementing deep

learning in BRB fault detection, suitable features are extracted and used for training the deep learning model. After the training, the trained model is used for predicting the signal from a motor. These procedures are summarized in Fig. 1(a). As mentioned, a large dataset is required to learn features in the deep learning method. However, with the limited dataset, feature extraction is still required and spectral-based features are necessary to effect deep learning improvements, as shown in a previous work.[3] The feature extraction based on fast Fourier transform (FFT), the BRB fault peak, and the envelope of a signal are described in Sects. 2.2–2.4, respectively.

2.2 Fast Fourier transform

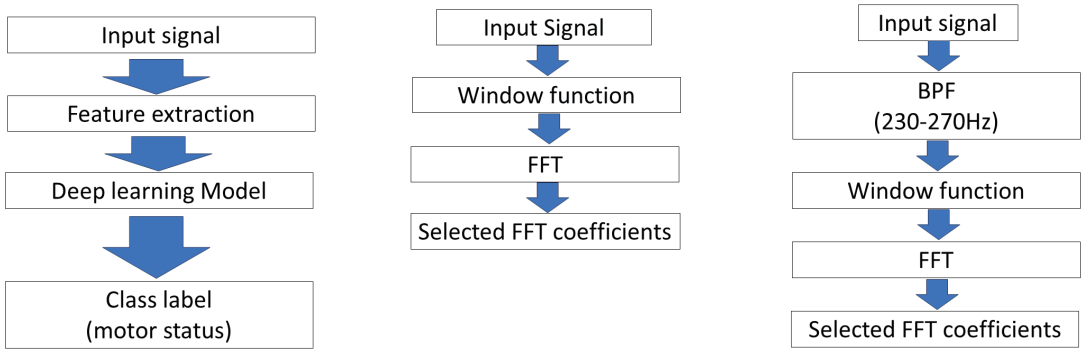
FFT is an algorithm for calculating the discrete Fourier transform with reduced computational complexity. To extract the spectral feature, the time domain signal is captured and transformed to the frequency spectrum using FFT. In this spectrum, frequency peaks are used in the analysis. The quality of the frequency peak separation depends on the frequency resolution[1,10] as shown in

$$f_{res} = f_{sampling} / N_{fft} , \tag{1}$$

where f_{res} is the frequency resolution, $f_{sampling}$ is the sampling frequency, and N_{fft} is the sample length of FFT.

2.3 BRB fault peak

To detect faults by the MCSA method, the peak of the frequency spectrum needs to be calculated. Various faults in an induction motor have a specific frequency spectrum that can be



(a) (b) (c)

Fig. 1. (Color online) Deep learning BRB fault detection: (a) overall detection procedures, (b) FFT spectral feature, and (c) signal envelope features.

determined using specific formulae. As previously mentioned, in this paper, we focus only on the BRB fault. In a previous work,[3] the BRB fault frequency peaks were determined using

$$f_{\mathrm{Low}} = (v - 2ks)F_{\mathrm{supply}}, \tag{2}$$

$$f_{\mathrm{Up}} = (v + 2ks)F_{\mathrm{supply}}, \tag{3}$$

where f_{Low} is the lower fault frequency component, f_{Up} is the upper fault frequency component, v is the v^{th} harmonic of the power supply frequency, k is the k^{th} harmonic of fault components, s is slip, and F_{supply} is the power supply frequency.

These calculation results are used to select the peak on the basis of the FFT coefficients. The overall process of the FFT spectral feature extraction is shown in Fig. 1(b).

2.4 Envelope of signal

The envelope of the signal generated from the Hilbert transform can also be used in fault diagnosis,[3,9,16] However, the envelope of the signal is in the time domain, so the FFT of the envelope should be performed to feed the deep learning model. Since some power supplies can have many types of noise, the envelope at a high harmonic can be used. As shown in the reference work,[3] signals in the sub-band between 230 and 270 Hz are selected via a band pass filter (BPF); therefore, only the components around the fifth harmonic are kept for calculating the signal envelope. The overall signal envelope feature extraction process is shown in Fig. 1(c).

2.5 Layer configuration

Basically, the deep learning model is composed of the convolution layer (Conv) and fully connected layer (FCL).[13] The numbers and sizes of these layers are the factors affecting both the training and prediction performance. Since the model with multiple layers has many parameters to be tuned, a powerful computer is required.

3. Methods

Here, we will explain how to modify the features and deep learning architecture. Our method uses FFT information from some particular part of the frequency spectrum to generate features, and we propose to use a modified deep learning architecture that is more suitable for these FFT features. From the results of the analysis of *spectral information*, we found that the feature extraction process in the reference method[3] can be modified to reduce its complexity, but only serial-structure deep learning can be used. In the following sections, we will describe how the proposed reduction can be performed.

3.1 Feature extraction by FFT only

The reference deep learning method provides the dataset[3] online with three class labels: healthy, 1-hole broken bar, and 2-hole broken bar (2BRB). Thus, basic statistical analysis can be applied to see differences in spectral data. After averaging all the FFT spectra in the same fault class, three spectral patterns shown in Fig. 2 are compared. Since the spectral pattern for each class is unique and its fault peaks are not fully overlapped (around 45 and 55 Hz), it is not necessary to perform the FFT spectral feature extraction from the signal envelope for further analysis.

3.2 Modified feature extraction

As previously described in Sect. 2.3, the feature from the selected relevant fault peaks found in the sample spectra can be used if the location of the peak is clear. However, it is difficult to select the correct peaks when other types of fault or supply harmonics produce overlapping peaks. In the reference work,[3] a 9.3 s duration was used to compute FFT, and sub-bands around the peaks were also used. However, the locations of the peaks need to be identified beforehand. Therefore, the proposed method includes all FFT coefficients corresponding to frequencies between f_{Low} and f_{Up}, including the power supply and its harmonic peaks, except the exact location of the correct fault peaks. The following steps are taken to extract the proposed features. (1) Find the FFT spectrum using Eq. (4) and set v=1 for the first sub-band.

$$FFT_{spec}(X) = |FFT(x)| \tag{4}$$

$FFT_{spec}(X)$ is the FFT spectrum, and $FFT(x)$ is FFT of the x signal in the time domain.

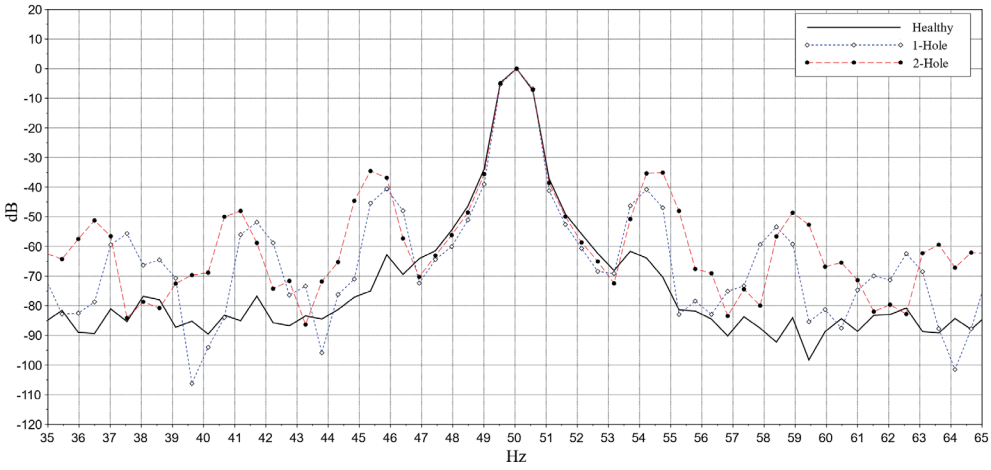


Fig. 2. (Color online) Healthy, BRB, and 2BRB fault spectra.

(2) Calculate f_{low} and f_{up} using Eqs. (2) and (3).
(3) Calculate FFT frequency bins b corresponding to f_{low} and f_{up} using

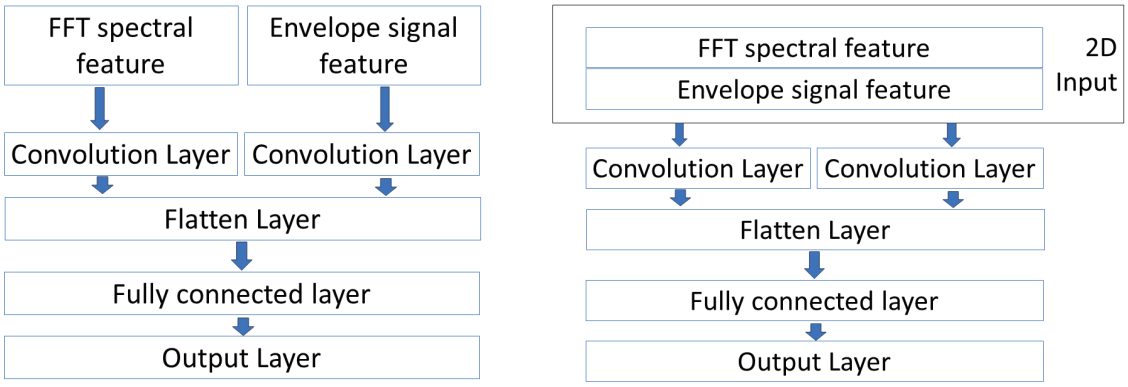
$$b = round\ (f/f_{res}) + 1. \tag{5}$$

(4) Keep the spectral coefficients between the frequency bins calculated in step (3). These coefficients are spectral components for the vth sub-band.
(5) Set $v = v + 1$.
(6) Repeat step 2) until $v = 5$.
(7) Generate the feature vector by concatenating all spectral components from the 1st to 5th sub-bands.

We set s, F_{supply}, and k to 0.05, 50 Hz, and 1, respectively. This makes the frequency of each sub-band 10 Hz. As five sub-bands are used to extract the features, a total bandwidth of 50 Hz is used. The input signal is preprocessed by the Hann window function before FFT. Since the feature vector has a length of 96 in to enable a comparison with the reference work,[3] the FFT resolution is set to around 0.52 Hz. After calculation using Eq. (1) with the sampling period T_s of 0.0007 s, 2,740 FFT points (pts) are needed for feature extraction. This FFT length takes 0.2 s, which is a shorter section duration than that in the reference work,[3] leading to a shorter time interval for reporting the detection result.

3.3 Deep learning architecture

The deep learning with parallel structure used in the previous work[3] can produce a good prediction result but increases the complexity of the model. This model is called double-input convolutional neural network (DICNN) and is shown in Fig. 3(a). Two separated 1D input layers are used to receive FFT and the signal envelope features separately. All the convolution layers are processed in 1D format. The good result of this method may be attributable to the increase in the number of parameters.



(a) (b)

Fig. 3. (Color online) Parallel structure models. (a) DICNN and (b) PCNN.

Alternatively, for the proposed method, the hypothesis is that deep learning with only a serial structure may provide the same results if the features and model are carefully designed.[14] Although 1D convolution requires less computation, 2D convolution is widely compatible with many computer devices, and 2D convolution operation has the option to work as 1D.[17] Therefore, in this work, we propose the 2D parallel convolution neural network (PCNN) using the 2D input layer. Both features are processed together, as shown Fig. 3(b). Moreover, for comparison with the serial structure, the common 2D convolution neural network (CNN) shown in Fig. 4(a) is used in the experiment as well. Finally, the use of the single FFT spectral feature together with the small convolution neural network (SCNN) shown in Fig. 4(b) is proposed.

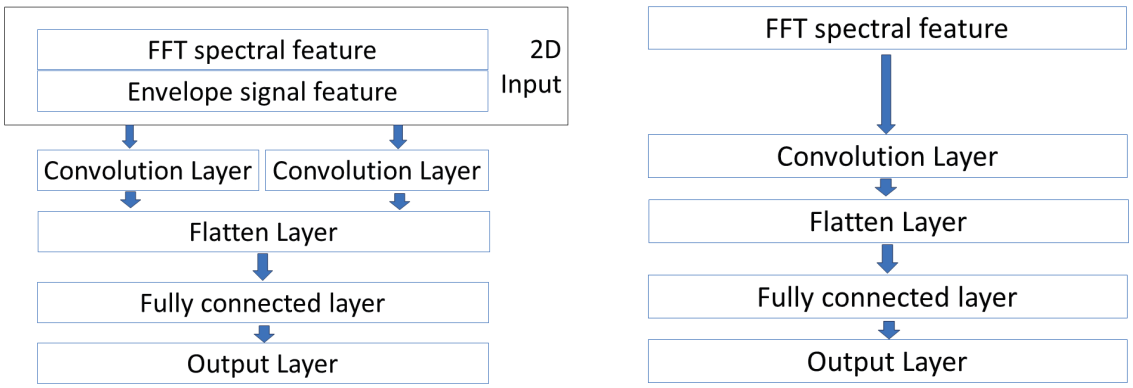
4. Experiment

The experiment was set up to verify the proposed method. The performance of the proposed method is compared with that of the reference method.[3] All the experiments were run on an Intel core i3-4160 computer without a GPU. The details of the experiment are discussed in the following subsections.

4.1 Dataset

The dataset used in this research is from LIAS laboratory[18] since it is used in the reference deep learning method.[3] The dataset consisted of the recorded input variables of a three-phase 1.1 kW induction motor. The sampling period of the recorded data was 0.0007 s and the data was recorded within 9.3 s.

In the reference work,[3] the distribution of the class labels was uneven as some classes had only eight examples whereas other classes had 18 examples. Thus, the results reported in the reference work are considered to be a case of class imbalance. To compare the performance of the dataset when all class labels are balanced, in this research, we included an additional dataset with a balanced class configuration in the experiment.



(a) (b)

Fig. 4. (Color online) Serial structure models. (a) CNN and (b) SCNN.

An additional configuration of the limited-balance class labels has eight examples per class for all class labels. Therefore, the number of samples is less than that in the original dataset. A summary of the dataset configurations is shown in Table 1. Since the proposed method uses a smaller FFT length than that for recording samples in the original dataset, each record is split into multiple small sections of 2740 sample points corresponding to the FFT length. As in the reference work,[3] training and testing sets were divided equally.

4.2 FFT length

The FFT length can affect both the performance and complexity of the trained model. Reduction of the FFT length can lead to a small number of feature vectors, resulting in less information owing to low frequency resolution, as seen in Eq. (1). To visualize this affect, we included a small feature having only 82 coefficients extracted from 2304 FFT points. The neural network model used with this reduction feature in the experiment is the same as the SCNN, but it is labeled as SSCNN to distinguish the smaller feature length.

4.3 Deep learning model parameters

The deep learning model architecture, previously described in Sect. 3.3, is used in the experiment. The parameters for different layers are shown in Table 2. The last row of the table contains the total training parameter (Param), and it can be used to compare the complexity of

Table 1
Dataset configuration.

	Healthy		1 Hole		2 Holes	
Speed	All	Limited	All	Limited	All	Limited
Low	18	8	8	8	8	8
Mid	18	8	18	8	8	8
High	18	8	18	8	18	8
Total	54	24	44	24	34	24

Table 2
Layer configurations.

	DICNN		PCNN		CNN		SCNN		SSCNN	
	Filter		Filter		Filter		Filter		Filter	
Layer	Size	Number	Size	Number	Size	Number	Size	Number	Size	Number
Input	2 × 96	–	2 × 96	–	2 × 96	–	96	–	82	–
Conv1	1 × 4	20	2 × 4	20	2 × 4	20	1 × 24	3	1 × 20	3
Conv2	1 × 10	16	1 × 10	16	1 × 10	16	–	–	–	–
Conv3	1 × 16	12	1 × 16	12	1 × 16	12	–	–	–	–
Conv4	1 × 20	8	1 × 20	8	1 × 20	8	–	–	–	–
Conv5	1 × 32	4	1 × 32	4	1 × 32	4	–	–	–	–
FCL	256	–	256	–	256	–	12	–	12	–
FCL	64	–	64	–	64	–	3	–	3	–
FCL	3	–	3	–	3	–		–	–	–
Param.	74523		74683		27579		2754		2382	

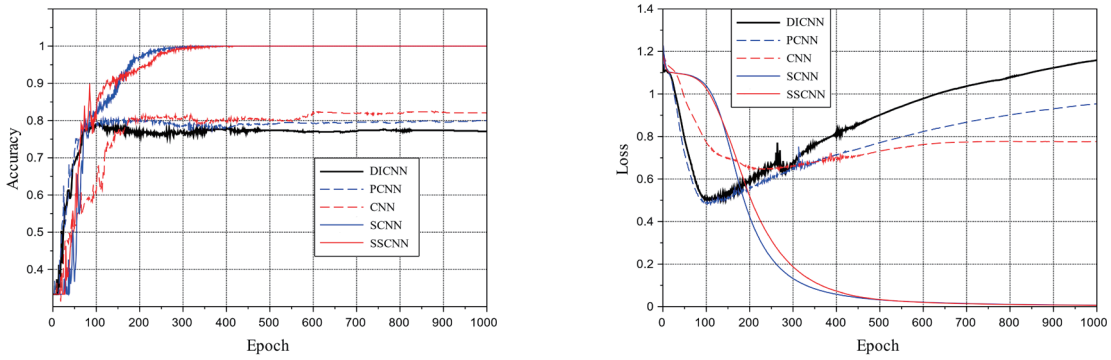
the model. For all models using the features generated by both FFT and Hilbert transform, the parameters are almost the same as in the original DICNN in the reference work[3] except that PCNN and CNN use a 2D filter in the Conv1 layer. However, SCNN and SSCNN are models using the features from the FFT spectrum only and need fewer parameters than the original DICNN. Both SCNN and SSCNN parameters are determined during the experiment.

4.4 Training parameters

The parameters used by the training algorithm were determined during the experiment. Following the methodology presented in the literature,[14] all the networks were trained and tested five times. Thus, all the results reported are averaged values. This is because the nature of training algorithms is stochastic; thus, the result from a single training phase may not reflect the best performance or may not be a fair comparison with other models. The software for training the model is Keras, which is implemented using Python language. The ratio of training to test sets is 1:1, which is the same as reported in the reference model.[3]

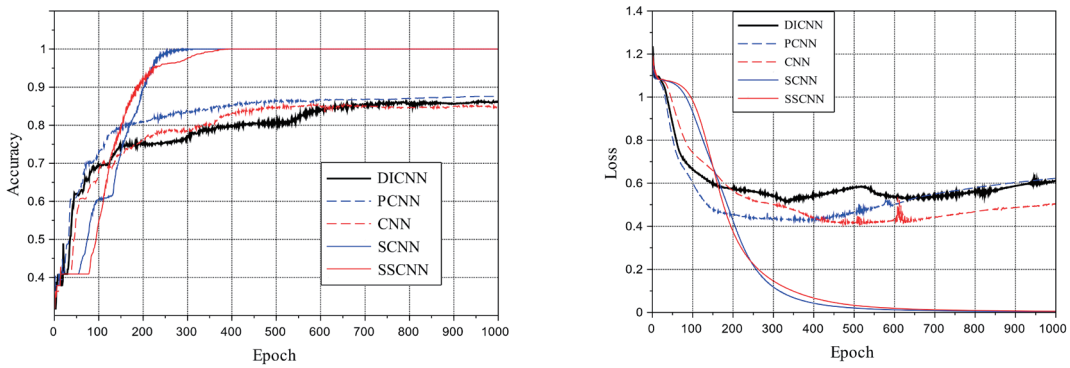
5. Results and Discussion

All training performances on the test set in terms of both accuracy and loss values are shown in Fig. 5 for the experiment on the limited-balance samples and in Fig. 6 for the experiment on all test samples. The results are summarized in Table 3. The accuracy results of most machine learning techniques presented in many studies are close to 1.00,[2,3,11] but the comparison in this accuracy metric alone is not useful. Moreover, those studies were usually conducted using only their own datasets that are not publicly available for fair comparison. Therefore, in our research, we focus only on other aspects such as feature size, balance class performance, and layer configurations, while using the same dataset available online. The main findings will be discussed in terms of features, architecture, and model complexity reduction.



(a) (b)

Fig. 5. (Color online) Training performance on limited-balance samples. (a) Accuracy and (b) loss.



(a) (b)

Fig. 6. (Color online) Training performance on all test samples. (a) Accuracy and (b) loss.

Table 3
Summary of results.

		DICNN(3)	PCNN	CNN	SCNN	SSCNN
Limited	Train	1.00	1.00	0.98	1.00	1.00
	Test	0.77	0.79	0.81	1.00	1.00
All	Train	0.99	1.00	0.99	1.00	1.00
	Test	0.85	0.87	0.84	1.00	1.00
Sample length		2470	2470	2470	2470	2304

5.1 Features

The features extracted from FFT are adequate to distinguish the differences between fault classes. From Table 3, it is shown in the results of SCNN and SSCNN that the model without Hilbert transform can show better performance than DICNN used in the reference work.[3] Although the feature size is reduced in SSCNN, the accuracy on the test sets is still 1.00. However, for the models using both FFT and Hilbert transform features, the results show good training accuracy on both the limited dataset and all datasets, but these models have test accuracies lower than 0.9 on both types of dataset. This might suggest that using features from Hilbert transform in this BRB detection problem leads to model overfitting. The number of samples used for training can affect the accuracy of the model, as shown in previous studies.[13,19] Thus, the larger number of samples from all test samples might be the reason why the models trained with this type of dataset have slightly better test accuracy than the models trained with the limited dataset.

5.2 Architecture

The reference work using the deep learning method[3] is reported to perform detection at 1.00 accuracy. However, there are no implementation details of how two 96-length inputs are created, and there is no information on how training and testing samples were selected during the

original experiment. Therefore, we compare the results on the deep learning architecture implemented in our new experiment setup to ensure that the same training and testing samples are used. From the results, DICNN, PCNN, and CNN perform similarly in terms of accuracy at 0.85, 0.87, and 0.84, respectively, on the original imbalanced dataset. The results also show that the accuracy of these models is lower on the limited-balance dataset, and the performance achieves constant accuracy at around 300 epochs, as shown in Fig. 5(a). The larger number of samples from all the original datasets cause the accuracy of these models to further increase after 300 epochs, as shown in Fig. 6(a). This effect is also clearly apparent in Figs. 5(b) and 6(b), that is, the loss is not improved after around 100 epochs.

5.3 Reduction of model complexity

From the results, our carefully designed model with reduced complexity for BRB detection can provide better performance than that of the reference work.[3] SCNN and SSCNN require fewer training parameters than DICNN, PCNN, and CNN, but they have higher accuracy. This is because the deep learning model can be overfitted to some examples presented in the training set if the model has too many parameters. Moreover, models with many parameters lead to calculation complexity for the computer. Thus, the small model with only the necessary parameters should have a better prediction performance. The training performance also shows that the performance increases after 100 epochs for both dataset configurations, as shown in Figs. 5(b) and 6(b).

6. Conclusions

In this work, we presented BRB fault detection using deep learning techniques. The results of analysis and experiment showed that the modification of the spectral feature from only FFT can provide adequate information for training the model, and it is not necessary to use additional feature vector extraction by Hilbert transform. The models using feature vectors from only FFT have better accuracy than the original models. This is because, with the same number of training set, the models with a large number of parameters can lead to overfitting. Apart from accuracy improvement, the proposed feature extraction uses a shorter duration from the sampled signal compared with the reference method. This can reduce the time and the possibility of motor damage. However, the method has a shortcoming when the signal has noise produced from the electrical system or other fault peaks from other fault types occur in combinations. We will investigate the development of a noise tolerance feature to be used with the deep learning method in the future.

Acknowledgments

This work was supported by the School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand.

References

1 P. Gangsar and R. Tiwari: Mech. Syst. Signal Process. **144** (2020) 106908. https://doi.org/10.1016/j.ymssp.2020.106908
2 R. F. Ribeiro Jr., I. A. D. S. Areias, M. M. Campos, C. E. Teixeira, L. E. B. D. Silva, and G. F. Gomes: Measurement **190** (2022) 110759. https://doi.org/10.1016/j.measurement.2022.110759
3 A. Boushaba, S. Cauet, A. Chamroon, E. Etien, and L. Rambault: Sensors **22** (2022) 9494 https://doi.org/10.3390/s22239494
4 A. M. Younus, and B. -S. Yang: Proc. 2010 Prognostics and System Health Management Conf. 1–6. https://doi.org/10.1109/PHM.2010.5414573
5 K. N. Gyftakis, A. J. M. Cardoso, and J. A. Antonino-Daviu: Mech. Syst. Signal Process. 1**93** (2017) 30. https://doi.org/10.1016/j.ymssp.2017.01.046
6 S. Halder, S. Bhat, D. Zychma, and P. Sowa: Energies **15** (2022) 8569. https://doi.org/10.3390/en15228569
7 S. Kumar, D. Mukherjee, P. K. Guchhait, R. Banerjee, A. K. Srivastava, D. N. Vishwakarma, and R. K. Saket: IEEE Access **7** (2019) 90690. https://doi.org/0.1109/ACCESS.2019.2926527
8 D. T. Hoang and H. J. Kang: IEEE Trans. Instrum. Meas. 69, **6** (2020) 3325. https://doi.org/10.1109/TIM.2019.2933119
9 J. Rangel-Magdaleno, H. Peregrina-Barreto, J. Ramirez-Cortes, and I. Cruz-Vega: Measurement **109** (2017) 247. https://doi.org/10.1016/j.measurement.2017.05.070
10 P. Pillay and Z. Xu: Proc. 1996 IEEE 31st Industry Applications Conf. (IEEE, 1996) 587–594.
11 W. Zhao, Y. Lv, J. Xiao, and Y. Li: Proc. 2021 Global Reliability and Prognostics and Health Management 1–6.
12 K. E. Bouchefry and R. S. de Souza: Knowledge Discovery in Big Data from Astronomy and Earth Observation, P. Škoda and F. Adam, Eds. (Elsevier, 2020) Chap. 12. https://doi.org/10.1016/B978-0-12-819154-5.00023-0
13 I. H. Sarker: SN Comput. Sci. **2** (2021) 1. https://doi.org/10.1007/s42979-021-00815-1
14 X Chen, B. Xu, and H. Lu: Proc. 2020 The Int. Conf. Communications, Information System and Software Engineering (CISSE 2020) 1–6. https://doi.org/10.1088/1742-6596/1792/1/012074
15 H. Schulz and S. Behnke: Künstliche Intelligenz **26** (2012) 357. https://doi.org/10.1007/s13218-012-0198-z
16 V. C. M. N. Leite, J. G. B. da Silva, G. L. Torres, G. F. C. Veloso, L. E. B. da Silva, E. L. Bonaldi, and L.E. L. de Oliveira: Bearing Technology, P. H. Darji, Eds. (InTech, 2017) Chap. 5. https://doi.org/10.5772/67145
17 Tensorflow documentation: https://www.tensorflow.org/api_docs/python/tf/nn/conv1d (accessed July 2023).
18 LIAS dataset: https://www.lias-lab.fr/mcsa (accessed July 2023).
19 D. Rajput, W. J. Wang, and C. C. Chen: BMC Bioinf. **24** (2023) 1. https://doi.org/10.1186/s12859-023-05156-9