Intelligent Bearing Fault Diagnosis with Convolutional Long-Short-Term-Memory Recurrent Neural Network

Amin Khorram\textsuperscript{a,*}, Mohammad Khalooei\textsuperscript{b}

\textsuperscript{a}Dept of Mechanical Engineering, WSE Co., Bozorgmehr St, 1416844816 Tehran, Iran.
\textsuperscript{b}Dept of IT and Computer Engineering, Amirkabir University of Technology, Hafez st, 15875-4413 Tehran, Iran.

\textbf{ARTICLE INFO}

\textbf{Article history:}
Received 00 December 00
Received in revised form 00 January 00
Accepted 00 February 00

\textbf{Keywords:}
Intelligent Fault Diagnosis
Bearing Fault
Intelligent controller
CNN+LSTM
Deep learning
IMS bearing dataset
CWRU bearing dataset

\textbf{ABSTRACT}

Fault diagnosis is an important topic both in practice and research. There is an intense pressure on industrial plants to continue reducing unscheduled downtime, performance degradation, and safety hazards, which requires detecting and recovering of potential fault in its early stages. Intelligent fault diagnosis is a promising tool due to its ability in rapidly and efficiently processing collected signals and providing accurate diagnosis results. In the literature, although many studies have developed algorithms for detecting bearing fault, the results have generally been limited to relatively small train/test datasets and the input data has been manipulated (selective features used) in order to reach a high accuracy. In the following paper a Convolutional Long-Short-Term-Memory Recurrent Neural Network (CRNN) is proposed for intelligent fault diagnosis of bearings. The purpose is to introduce an algorithm which takes directly the raw time-series sensor data as the input and detects the health condition of the bearing as the output with a high accuracy and in a short period of time. The method can reach the highest accuracy to the best knowledge of author of the present paper voiding any sort of pre-processing or manipulation of the input data. The paper starts with a brief description of the new approach of diagnosis using a CRNN network, which the author plans to develop and implement for diagnosing bearing faults and concludes with the identification of the most significant advantages of the proposed method as well as a comparison between the proposed method and the other methods in the literature.

1. Introduction

Bearings are the key components in rotary machines. Bearing fault is one of the main reasons for motor failure and to detect the fault in primary stages can prevent a great down-time and recovery cost (Bonnett and Yung 2008). In recent years implementation of machine learning/deep learning techniques in many scientific fields has been drastically increased. Intelligent fault detection is one of those areas which has received a wide attention and has been used in practical situations. The key issue of applying machine learning techniques into bearing fault diagnosis is developing a network architecture which can get satisfactory diagnosis performance in a relatively short time (Mao et al. 2019). Data-driven intelligent fault detection of bearing is mainly conducted using signal processing. The signal of which is driven from acceleration sensors: “vibration signal”, or the signal driven from frequency inverters: “motor current signal” (Lessmeier et al. 2016). In literature, vibration signal has received more attention due to the more accurate results. In order to implement machine learning/deep learning techniques for bearing fault detection we need to extract features and use the features in a learning algorithms aiming to reach the highest accuracy. Features can be obtained using time domain (Zhou et al. 2008), frequency domain (Schoen et al. 1995) or time-frequency domain (L. Eren and Devaney 2004). In the past decade machine learning techniques represented by k-nearest-neighbour (KNN), support-vector-machine (SVM) and artificial-neural-network (ANN) has been promising tools for bearing fault detection (Mao et al. 2019). However, the output of machine

* Corresponding author.
E-mail address: amin.khorram@rockets.utoledo.edu
learning techniques are typically acceptable in case of relatively small-scale-data (Samanta and Nataraj 2009). Therefore, all the past efforts in bearing fault diagnosis using conventional machine learning methods were accomplished on relatively small datasets and by using complex data pre-processing techniques in order to extract the most detectable features. For instance, in (F. Yaqub et al. 2012) Yaqub et al. used KNN for bearing fault diagnosis and tested a small data-frame he also used higher-order-cumulants (HOC) and wavelet-transform (WT) for pre-determined transformation of data, nevertheless, did not reach an acceptable accuracy. In (Hu et al. 2007) Q et al. used SVM and again the data was pre-processed and the size of data-frame was relatively small. Evidently, with the quick development of advanced measurement techniques, massive data are collected and most of the conventional machine learning algorithms have drawbacks to establish decision models on these data (Zhang et al. 2017). Hence, in recent years tendencies have shifted from conventional methods to more complex ones such as deep neural network (DNN), convolutional neural network (CNN) and recurrent neural network (RNN). In (Levent Eren et al. 2018) Eren et al. utilizes one dimensional convolutional neural network for time series prediction, however, the input data is filtered-decimated and normalized in order to reach a better efficiency. In (Zhang et al. 2017) Zhang et al. claims to reach a high accuracy using DNN for time sequence prediction, however, the article does not provide any architecture for the DNN network. The same story happens in (Mao et al. 2019) where Mao et al. claims to reach a high accuracy using a novel deep learning method although, does not provide any feasible architecture for their proposed network. In this work we are going to use a CNN+LSTM network for temporal sequence prediction of the data obtained in time domain, aiming to reach the highest accuracy in a relatively short time. Compared to the other articles in the literature, we are not doing any pre-processing or manipulation on the raw data. As a result, the model can be utilized in practical situation and extract the real characteristic of the practical system’s signal under all circumstances. In the following section we are going to explain the method and our proposed CRNN architecture. In the next section we are going to evaluate our proposed model by testing two benchmark datasets in the literature: Intelligent Maintenance Systems (IMS) bearing dataset (J. Lee 2007) which is a run to failure raw bearing dataset measured by Centre of Intelligent Maintenance Systems of University of Cincinnati and Case Western Reserve University (CWRU) bearing dataset (Case Western Reserve University 2017) measure by Bearing Data Centre of Case Western Reserve University. The features have been obtained in time domain and using CRNN, the model is actually predicting time sequences. Therefore, the dataset is supposed to be divided into tensors of equal size, representing a temporal sequence each. In this article we are going to provide the architecture and the step by step path we went through, in order to reach a high accuracy for our proposed fault detection methodology.

The contribution of the following paper can be summarized as follows:

1-Using a relatively bigger data-set for training and testing, compared to the other papers in the literature and reaching a higher generalization accuracy at least possible time (relatively less time and number of epochs in our proposed deep learning model, compared to the articles in the literature).

2-A new deep learning structure is proposed for bearing fault diagnosis by integrating CNN-LSTM. By applying the proposed deep learning method, this paper can effectively utilize the time series to improve the diagnosis accuracy and numerical stability for bearing fault.

3-The model can be fed by the raw vibration data directly and no pre-processing and pre-determined transformation (such as FFT or DWT), manipulated feature extraction and feature selection is required.

The paper is organized as follows. In section 2, a brief review to CNN-LSTM and the network architecture used in this work. In section 3, we describe the test rigs, the datasets, the fault classification and our proposed CRNN architecture as well as computer experiments and analysis. Section 4 is devoted to discussion and comparison of our method with the other methods in the literature and finally section 5 represents the conclusion and future works.

2. Method

2.1 CNN+LSTM network

CNNs are biologically inspired feed-forward ANNs which are considered as simple computational models of the mammalian visual cortex (Alex et al. 2012). Therefore, they are mainly used for 2D signals such as images and video. Nowadays CNNs are widely used in the machine vision community as a state of the art technology for many image and video recognition problems (Levent Eren et al. 2017). A complete CNN stage contains a convolution layer and a pooling layer. The value of a neuron \( v_{ij} \) at position \( x \) of the \( j \)th feature map in the \( i \)th layer is denoted as follows:

\[
v_{ij}^x = g \left( b_{ij} + \sum_{m=0}^{p-1} \sum_{p=0}^{n^{-1}} w_{ijm}^p v_{(i-1)m}^{x+p} \right)
\]  

(1)

\[
g(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]  

(2)

where \( m \) indexes the feature map in the previous layer, \((i-1)_m\) layer connected to the current feature map, \( w_{ijm}^p \) is the weight of position \( p \) connected to the \( m \)th feature map, \( P \) is the width of the kernel toward the spectral dimension, and \( b_{ij} \) is the bias of \( j \)th feature map in the \( i \)th layer. Pooling can offer invariance by reducing the resolution of the feature maps (Zhen Zuo 2016). Each pooling layer corresponds to the previous convolutional layer. The neuron in the pooling layer combines a small \( N \times 1 \) patch of the convolution layer. The most common pooling operation is max pooling (Chen et al. 2016):

\[
a_j = \max_{n=1}^{N_x} u(n,l)
\]  

(3)

Where \( u(n,1) \) is a window function to the patch of the convolution layer, and \( a_j \) is the maximum in the neighbourhood.

Long Short-Term Memory (LSTM) networks are recurrent neural networks equipped with a special gating mechanism that controls access to memory cells (Hochreiter and Schmidhuber 1997). Since the gates can prevent the rest of the network from modifying the contents of the memory cells for multiple time steps, LSTM networks preserve signals and propagate errors for much longer than ordinary recurrent neural networks. LSTM was designed to model
temporal sequences and their long-range dependencies more accurately than conventional RNNs (Sak 2014). For brevity, we omit the rather extensive equations describing the LSTM network and illustrate it an LSTM block in Fig.1.

![LSTM memory cell with gating units](image)

**Fig.1.** LSTM memory cell with gating units

A standard neural network unit i only consists of the input activation a, and the output activation b, which are related when a tanh activation function is used:

\[ b_i = \tanh(a_i) \]  

(4)

The LSTM unit adds several intermediate steps: After applying the activation function to input a, the result is multiplied by a factor b. Then the inner activation value of the previous time step, multiplied by the quantity b, is added due to the recurrent self-connection. Finally, the result is scaled by b and fed to another activation function, yielding b. The factors b, b, b ∈ (0, 1), indicated by the small white circles, are controlled by additional units (depicted as blue circles) called input, output, and forget gate, respectively. The gating units sum the activations of the previous hidden layer and the activations of the current layer from the previous time step as well as the inner activation of the LSTM unit. The resulting value is squashed by a logistic sigmoid function which then is set to b, b, or b, respectively(Sundermeyer et al. 2012). The extensive equation for LSTM network can be observed in(Graves and Schmidhuber 2005).

CNN-LSTMs are used for many visual learning tasks but are also known to be used for speech recognition and natural language processing(Bilgera et al. 2018). Moreover, CNNs and LSTM are both powerful tools for temporal/spatial sequence prediction(Yao et al. 2018). Dealing with big data or complex temporal/spatial sequence problems, CNN-LSTM network enhances accuracy and precision of predictions(Huang and Kuo 2018).

In order to explain the temporal/spatial sequence prediction, suppose we observe a dynamical system over a temporal/spatial region represented by an MxN grid which consists of M rows and N columns. Inside each cell in the grid, there are P measurements which vary over time. Thus, the observation at any time can be represented by a tensor X∈R^MxNxP, where R denotes the domain of the observed features(Shi et al. 2015). If we assume that the observations are recorded periodically, we can divide the dataset into samples of equal temporal/spatial length and as a result we have a sequence of tensors \( \tilde{X}_t \), \( \tilde{X}_t \), \( \tilde{X}_t \), \( \tilde{X}_t \). The spatiotemporal sequence forecasting problem is to predict the most likely length-K sequence in the future, given the previous J observations which include the current one:

\[ \hat{Y}_{t+1:k} = \arg \max_{X_{t+1:k}} p(X_{t+1:k} | \tilde{X}_{t-J+1}, \tilde{X}_{t-J+2}, ..., \tilde{X}_t) \]  

(5)

### 2.2. Architecture and Learning method

In this study the implementation of a CNN-LSTM network is proposed for intelligent bearing fault diagnosis system. Our proposed CNN-LSTM architecture is depicted in Fig.2. The raw vibration datasets or features have been collected in time domain and as we know, CRNN network is supposed to be fed by tensors of equal size. Therefore, the first step is to divide the dataset into samples or temporal sequences of equal length in order to feed them to our CRNN model. In the next step, the features are split into train, validation and test sets. Our deep neural network consists of a one-dimensional convolutional neural network with 84 filters and kernel-size of 84 and an LSTM layer with 24 neurons (the reason why we chose this architecture is going to be explained in section 3). There is one dropout layer after each main layer in order to prevent overfitting(Srivastava et al. 2014). We have also used batch-normalization layers in order to speed up and enhance the stability of network and the accuracy of learning(Ioffe and Szegedy 2015). Finally the fully-connected layer takes advantage of sigmoid activation. The loss and optimizer functions used for compiling are mean-squared-logarithmic-error (MSLE) and adagrad respectively. The CRNN network predicts \( \hat{y} \) and using the following loss function the deviation and accuracy of the network is measured:

\[ L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{n} (log(y_i + 1) - log(\hat{y}_i + 1))^2 \]  

(6)
3. Experimental Signal Processing

In this section we are going to evaluate our proposed method, by testing it on two benchmark bearing datasets: IMS and CWRU. In the following we are going to describe the test rigs used for capturing the vibration data, fault classification (labels) and the raw vibration signals (features) which are used as the input of our CRNN algorithm. The section proceeds toward analysing the amplitude/time diagrams of the datasets, the procedure of designing our architecture and finally the accuracy/loss diagrams and confusion matrices for each test.

3.1. IMS Bearing Dataset

In order to validate the proposed method, experimental data are applied to test its performance. The dataset is provided by the University Of Cincinnati Center Of Intelligent Maintenance Systems [8]. The experimental apparatus is shown in Fig.3.

As depicted in Fig.3, there was a shaft on which four bearings were installed. The bearings model was Rexnord ZA-2115 double-row. Two high precision accelerometers were connected to each bearing in Cartesian coordinates, therefore, the vibration was measure in X and Y directions for each bearing. The shaft was driven by an alternative current (AC) motor which was connected to shaft using a conveyor belt. The shaft and bearings were under a radial load of 2721.5 Kg imposed by a spring mechanism. Rotation speed of the shaft was 2000 revolution per minute (RPM). The sampling rate was set as 20 KHz and every 20480 data points (recorded in one second) were recorded in a single file. In every 5 or 10 minutes, the data were recorded and written in files while the bearings were rotating. Each test consists of 2156 files. Therefore the total number of data-points is 44,154,880 for each test. Previous works done
on the 1st-test of IMS bearing dataset shows that there are seven different states of health during the test (Qiu et al. 2006):

- Early (initial run-in of the bearings)
- Normal
- Suspect (the health seems to be deteriorating)
- Imminent failure (for bearings 1 and 2, which didn’t actually fail, but were prone to damage)
- Inner race failure (bearing 3)
- Rolling element failure (bearing 4)
- Stage 2 failure (bearing 4)

The vibration signals of some states are pretty close which cannot be distinguished by signal processing, therefore in order to reduce the calculation complexity and improve the performance of our learning algorithm, we chose the labels with the highest importance both in fault detection and practical situation (the other states underlie the following ones):

- Healthy (data taken from early and normal states)
- Suspected
- Inner Race failure
- Rolling element failure

As we mentioned before the number of data-points in the 1st-test is pretty big and it is so time and memory consuming to use this big dataset as the input of our learning algorithm. Moreover, using the big dataset enhances the risk of overfitting. Therefore, we randomly chose 30 files for each class or state of health. In the next step data is concatenated, labelled and prepared to be fed to the learning algorithm. Labels are 0-(Healthy), 1-(Suspected), 2-(Inner-race-fault) and 3-(Rolling-element-fault). As explained in the previous section we are using a CRNN network hence, the input is supposed to be sequence of tensors with equal dimensions. The sampling rate is 20KHz and the rotation speed is 2000RPM, so it can be calculated that there are 600 points per revolution (rotation period). The size of each sample is set to be a quarter of the rotation period, which is 150 rows of data. In each row of data we have the vibration data of bearings in X and Y directions. Therefore, there are 8 features in each row namely:

- \( X_1, Y_1 \) (measured by accelerometers connected on the first bearing)
- \( X_2, Y_2 \) (measured by accelerometers connected on the second bearing)
- \( X_3, Y_3 \) (measured by accelerometers connected on the third bearing)
- \( X_4, Y_4 \) (measured by accelerometers connected on the fourth bearing)

As a result, each sample is a tensor of (150x8x1) dimension and the input tensor for each health state has a dimension of (4096x150x8). The total number of samples is 16384 with 4096 samples for each health state. The number of samples for each class has can be observed in Table 1. The Amplitude/time diagram of the four health states has been depicted in Fig.4. As it can be observed, each health state has a specific vibration signal signature.

### Table 1 – Number of samples and class number for each health state, IMS dataset.

| State           | Number of samples | Class(Label) |
|-----------------|-------------------|--------------|
| Healthy         | 4096              | 0            |

![Raw Vibration signal for healthy label](image1.png)

![Raw Vibration signal for suspected label](image2.png)

![Raw Vibration signal for Inner-race-fault label](image3.png)

![Raw Vibration signal for Rolling-element-fault label](image4.png)

Fig.4. IMS Bearing Dataset, raw vibration signal for Healthy, Suspected, Inner-race-fault and Rolling-element-fault.
In the next step, data has been split into train, validation and test sets. In order to take advantage of a stateful LSTM network, the split and batch-size selection should be performed in a way so that the number of samples in train and test sets be integers and divisible by the batch-size. The number of samples for all the four classes are 16,384. Therefore, we allocated 25% of dataset to train and 75% to test and the optimum batch-size turned to be 64.

In order to select the best architecture for our CRNN, networks with different parameters have been tested, the result of which can be observed in Table 2. The number of epochs for all the accomplished tests has been set to 50 and the goal is to reach the highest training accuracy at the shortest possible time. The simulation model is based on Keras Tensorflow backend library in Python. The Processor is Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 1992 MHz, 4 Core(s), 8 Logical Processor(s) and Physical Memory (RAM) is 8GB.

In test no.2, the number of LSTM neurons has been modified to 24 and the accuracies has clearly. Test no.16 reveal that exaggerating the number of filters/kernel size for the Conv1D layer, not only deteriorates the network performance. In tests no.4, no.5, no.6 we tried a second Conv1D layer, a second stateful LSTM layer and both additional layers at the same time, respectively. As it can be observed, adding layers does not have a positive effect on the test accuracy. Changing the architecture was presented in Fig.2.

Table 2 – Finding the best accuracy in a relatively short calculation time.

| Test No. | Conv1D Filters | Conv1D Kernel-size | Convolutional Activation | First Dropout | Maxpooling Size | LSTM Neurons | Second Dropout | Keras Loss | Keras Optimizer | Fully-Connected Activation | Train Accuracy | Test Accuracy | Calculation Time(seconds) |
|----------|----------------|-------------------|--------------------------|--------------|-----------------|--------------|---------------|------------|----------------|--------------------------|----------------|---------------|--------------------------|
| 1        | 32             | 8                 | elu                      | 0.01         | 8               | 12           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9721         | 0.9138        | 74                       |
| 2        | 32             | 8                 | elu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9772         | 0.9294        | 92                       |
| 3        | 32             | 8                 | elu                      | 0.01         | 8               | 36           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9828         | 0.9224        | 100                      |
| 4        | 32 | 16     | 8 | elu | 0.01 | 8 | 24 | 0.01 | MSLE | adagrad | Sigmoid | 0.9735 | 0.9248 | 101 |
| 5        | 32             | 8                 | elu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9822         | 0.9146        | 102                      |
| 6        | 32 | 16     | 8 | elu | 0.01 | 8 | 24 | 0.01 | MSLE | adagrad | Sigmoid | 0.9803 | 0.9172 | 98 |
| 7        | 32             | 32                | relu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | RMSprop        | Sigmoid                  | 0.9983         | 0.9517        | 149                      |
| 8        | 64             | 32                | elu                      | 0.01         | 8               | 12           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9992         | 0.9502        | 244                      |
| 9        | 64             | 32                | elu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9977         | 0.9521        | 269                      |
| 10       | 64             | 32                | elu                      | 0.01         | 8               | 24 | 0.01 | MSLE | adagrad | Sigmoid | 0.9998 | 0.9529 | 538 |
| 11       | 64             | 64                | elu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9998         | 0.9641        | 484                      |
| 12       | 84             | 64                | elu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9999         | 0.9595        | 334                      |
| 13       | 84             | 84                | elu                      | 0.01         | 8               | 12           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 1.0000         | 0.9619        | 538                      |
| 14       | 84             | 84                | elu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 1.0000         | 0.9713        | 419                      |
| 15       | 96             | 96                | elu                      | 0.01         | 8               | 24           | 0.01          | MSLE       | adagrad        | Sigmoid                  | 0.9999         | 0.9666        | 680                      |
| 16       | 128            | 84                | elu                      | 0.01         | 8               | 24 | 0.01 | MSLE | adagrad | Sigmoid | 1.0000 | 0.9639 | 661 |
| 17       | 84             | 84                | elu                      | 0.01         | 8               | 24 | 0.01 | MSLE | adagrad | Sigmoid | 0.9999 | 0.9675 | 500 |
| 18       | 84             | 84                | relu                      | 0.01         | 8               | 24 | 0.01 | MSLE | RMSprop | Sigmoid | 0.9993 | 0.9563 | 601 |
| 19       | 84             | 84                | relu                      | 0.01         | 8               | 24 | 0.01 | MSLE | adagrad | relu     | 0.9994 | 0.9456 | 597 |
| 20       | 84             | 84                | sigmoid                   | 0.01         | 8               | 24 | 0.01 | MSLE | adagrad | sigmoid | 1.0000 | 0.9561 | 507 |
| 21       | 84             | 84                | elu                      | 0.01         | 8               | 24 | 0.01 | MSLE | adam    | sigmoid | 0.9999 | 0.9668 | 626 |
| 22       | 84             | 84                | elu                      | 0.01         | 8               | 24 | 0.01 | Cat-Cr-entropy | adagrad | Sigmoid | 1.0000 | 0.9641 | 539 |
The train and test accuracies/losses diagrams of the optimum result on IMS dataset could be observed in Fig.5, and the confusion matrix for this test can be observed in Fig.6.

![Fig.5. IMS Bearing Dataset, raw vibration signal for Healthy, Suspected, Inner-race-fault and Rolling-element-fault.](image)

![Fig.6. Confusion matrix of the optimum result on IMS dataset](image)

As it can be observed in Fig.6, the classifier has missed some predictions in classes 1 and 3 or the “suspected state” and “outer-race-fault state”. Returning back to Fig.4, the signal diagram of the two classes are pretty close and confusing for the CRNN model. However, considering that we have not used any data pre-processing or data selection/manipulation, the strength of the model in Health vs Fault state diagnosis for this test can be rated as suitable.

We have reached a high fault detection accuracy for IMS bearing dataset, implementing our CRNN algorithm. In the next step we are going to test our proposed model for the second benchmark bearing dataset.

### 3.2 CWRU Bearing Dataset

The CWRU dataset is provided by Case Western Reserve University Bearing Data-Center(Case Western Reserve University 2017). The test rig has been shown in Fig.7.
The test rig consists of a 2-hp Reliance Electric motor, bearing and fastened accelerometer, a torque transducer/encoder and a dynamometer. The tested bearing was SKF deep-groove ball bearings 6205-2RS JEM. Accelerometer was placed at the 12 o’clock position of the motor housing. Data was collected at 12 KHz for drive-end-bearing-experiment. Single point fault was introduced to the test bearing using electro-discharge machining with fault diameters of 0.53 mm at the inner raceway, rolling element and outer raceway. The approximate motor speed was 1750 rpm. There are 6 states of health for this test:

- Normal (Healthy)
- Ball Fault
- Inner-Race-Fault
- Outer-Race-Fault at 3 o’clock (Fault placed at the load zone)
- Outer-Race-Fault at 6 o’clock (Fault placed orthogonal to the load zone)
- Outer-Race-Fault at 12 o’clock (Fault placed orthogonal to the load zone)

We chose 121155 data-points for each state of health. Implementing our CRNN model, the input is supposed to be sequence of tensors with equal dimensions. Given the sampling rate of 12KHz and rotation speed of 1750rpm, there are approximately 411 points per revolution (rotation period). We choose the number of data-points in each sample to be 205 corresponding to half a revolution approximately, in order to reach the highest train/test efficiency. The number of data-points for each health state is also selected so that be divisible by the number of data-point per sample. Therefore, each sample is a tensor of (205x1x1) dimension and the input tensor for each health state has the dimension of (591x205x1).

The Amplitude/time diagram of the six health states has been illustrated in Fig.9. As it can be observed, each health state has a specific vibration signal signature.

| State                     | Number of Samples | Class(Label) |
|---------------------------|-------------------|--------------|
| Normal (Healthy)          |                   |              |
| Ball Fault                |                   |              |
| Inner-Race-Fault          |                   |              |
| Outer-Race-Fault at 3 o’clock |               |              |
| Outer-Race-Fault at 6 o’clock |               |              |
| Outer-Race-Fault at 12 o’clock |              |              |

Fig.8. CWRU Bearing Dataset, raw vibration signal for Healthy, Suspected, Inner-race-fault and Rolling-element-fault.
The samples of equal size were split into train, validation and test sets. In order to take advantage of a stateful LSTM network, the split and batch-size selection should be performed in a way so that the number of samples in train and test sets be integers and divisible by the batch-size. The number of samples for all the six classes are 3546, therefore, we allocated 50% of dataset to train and 50% to test and the optimum batch-size turned to be 197. In this test we used the optimum architecture of which we achieved in the previous test and as a result the accuracy of train and test for 50 epochs turned to be 1.0000 and 0.9977 respectively. The calculation time for 50 epochs was 61 second.

The train and test accuracies/losses diagrams of the test on CWRU dataset could be observed in Fig. 9, and the confusion matrix for the test can be observed in Fig. 10.

As it can be observed in Fig. 10, the classifier has almost predicted all the six classes correctly. It only missed some predictions in classes number 4 and 5 or the “outer race fault-6 o’clock” and “outer race fault-12 o’clock” labels. Although, returning back to Fig. 9, the signal diagrams of the two classes are pretty close and considering that both faults are on the outer race we can overlook the small miss. Taking into consideration that we have not used any data pre-processing or data selection/manipulation, the strength of the model in Healthy vs Fault state diagnosis for this test can be rated as exceptional.
4. Discussion

Comparison of classification accuracies with different fault detection methods using the same benchmark datasets is shown in Table 4.

| Classifier | Data pre-processing | IMS test accuracy | CWRU test accuracy |
|------------|---------------------|-------------------|--------------------|
| KNN (F. Yaqub et al. 2012) | HOCs and WT | - | 91.23% |
| SVM (Hu et al. 2007) | WP | 62.5% | 98.7% (4-classes and small number of samples) |
| SVM ensemble (Hu et al. 2007) | WP | - | 89.8%-100% (4-classes and small number of Samples) |
| SVM (Wang et al. 2019) | Statistical locally linear embedding | - | 77.8-94.1% (4-classes and small number of samples) |
| DNN with temporal coherence (Zhang et al. 2017) | - | 94.9% | 94.4% (provide no architecture for their claim) |
| Compact 1D CNN (Levent Eren et al. 2018) | Filtering-Decimation-Normalization | 93.9% | 93.2% |
| Deep output kernel learning (Mao et al. 2019) | - | - | -(provided the training accuracy not testing accuracy) |
| Our proposed CRNN | - | 0.9713 | 0.9977 |

In the literature, almost all the previous methods have used some sort of data-pre-processing. For instance, Filtering, higher order cumulants (HOCs), wavelet transform (WT), wavelet packet transform (WP). Then, the best set of features are selected from high dimensional extracted features by applying various dimension reduction techniques such as principal component analysis. For classification of the selected features, various classifiers have been used although, we can observe that over the time tendencies have shifted from simpler supervised learning models such as SVM and KNN to more complex learning models such as CNN and DNN. The main drawback about almost all these studies is that they have used selected data or manipulated features in order to increase accuracy. The manipulated features may not represent the characteristics of the practical system’s signal under all circumstances. Consequently, this will limit general applicability of those solutions. Moreover, extracting high dimensional features along with necessary post-processing or feature selection methods can significantly increase cost and computational complexity of the whole system [11]. The next point is that, although many studies have reported a very high classification accuracy, their results have generally been limited to relatively small train/test datasets. We used a relatively bigger data-frame of learning features compared to the other articles mentioned in Table 3, and reached a high accuracy, voiding the need for any data pre-processing or manipulation. In addition, some previous studies claimed reaching a very high accuracy although did not provide their proposed network’s architecture and the feasible way through the accuracy (Mao et al. 2019; Zhang et al. 2017). In this paper, the network architecture of which was used to achieve a high accuracy and the analysis behind choosing every single element of the proposed network was presented.

5. Conclusion

In this work, performance of a generic real-time induction bearing fault diagnosis based on a new supervised machine learning approach has been extensively studied. The intelligent system employs Convolutional Recurrent Neural Network classifier that is fed by raw time domain features which are reshaped in form of tensors of time sequences. By learning the features automatically from the raw bearing vibration data with the proper training, the model is able to diagnose the fault ideally and considering the big dataset, in a relatively short time. Implementing the proposed method in practical situation and industrial scale, has the following advantages compared to the other methods:

- Monitoring a larger and more comprehensive recorded data.
- Reaching a more accurate prediction at a relatively shorter time and number of epochs.
- The model can be fed by the raw vibration data directly and no data pre-processing, pre-determined transformation (such as FFT or DWT), manipulated feature extraction and feature selection is required.
- The calculation is more cost effective compared to some solutions containing data-pre-processing and some complex deep architectures in the literature.

The CRNN classifier based fault diagnosis system is tested for bearing fault diagnosis from two commonly used real benchmark vibration datasets the experimental results validate the effectiveness and feasibility of the CRNN classifier in fault diagnosis. The classifier achieved overall classification accuracies of 97.13% for IMS and 99.77% for CWRU bearing datasets. Classification results demonstrated that CRNN can learn highly discriminative features directly from the raw input sensor data.

5.1 Future work

Testing and evaluation of the model using the collected motor current signal instead of vibration signal as well as reducing the calculation time of the system will be the future work.
REFERENCES

Alex, K., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. 1097--1105.

Bilgera, C., Yamamoto, A., Savano, M., Matsukura, H., & Ishida, H. (2018). Application of Convolutional Long Short-Term Memory Neural Networks to Signals Collected from a Sensor Network for Autonomous Gas Source Localization in Outdoor Environments. Sensors, 18(12), 4484.

Bonnett, A. H., & Yung, C. (2008). Increased Efficiency Versus Increased Reliability. IEEE Industry Applications Magazine, 14(1), 29-36, doi:10.1109/MIA.2007.909802.

Case Western Reserve University, B. D. C. (2017).

Chen, Y., Jiang, H., Li, C., Jia, X., & Ghamisi, P. (2016). Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks (Vol. 54).

Claessens, B. J., Vrancx, P., & Ruelens, F. (2018). Convolutional Neural Networks for Automatic State-Time Feature Extraction in Reinforcement Learning Applied to Residential Load Control. IEEE Transactions on Smart Grid, 9(4), 3259-3269, doi:10.1109/TSG.2016.2629450.

Eren, L., & Devaney, M. J. (2004). Bearing damage detection via wavelet packet decomposition of the stator current. IEEE Transactions on Instrumentation and Measurement, 53(2), 431-436, doi:10.1109/TIM.2004.823323.

Eren, L., Ince, T., & Kiranyaz, S. (2018). A Generic Intelligent Bearing Fault Diagnosis System Using Compact Adaptive 1D CNN Classifier.

F. Yaqub, M., Gondal, I., & Kamrizzaman, J. (2012). Inchoate Fault Detection Framework: Adaptive Selection of Wavelet Nodes and Cumulant Orders (Vol. 61).

Graves, A., & Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM networks. In Proceedings, 2005 IEEE International Joint Conference on Neural Networks, 2005. 31 July-4 Aug. 2005 2005 (Vol. 4, pp. 2047-2052 vol. 2004). doi:10.1109/IJCNN.2005.1556215.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780, doi:10.1162/neco.1997.9.8.1735.

Hu, Q., He, Z., Zhang, Z., & Zi, Y. (2007). Fault diagnosis of rotating machinery based on improved wavelet package transform and SVMs ensemble (Vol. 21).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

J. Lee, H. Q., G. Yu, J. Lin (2007). IMS, University of Cincinnati. "Bearing Data Set", NASA Ames Prognostics Data Repository. NASA Ames Research Center, Moffett Field, CA.

Lessmeier, C., Kimotho, J., Zimmer, D., & Sextro, W. (2016). Condition Monitoring of Bearing Damage in Electromechanical Drive Systems by Using Motor Current Signals of Electric Motors: A Benchmark Data Set for Data-Driven Classification.

Mao, W., Feng, W., & Liang, X. (2019). A novel deep output kernel learning method for bearing fault structural diagnosis. Mechanical Systems and Signal Processing, 117, 293-318, doi:https://doi.org/10.1016/j.ymssp.2018.07.034.

Qiu, H., Lee, J., Lin, J., & Yu, G. (2006). Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. Journal of Sound and Vibration, 289(4), 1066-1090, doi:https://doi.org/10.1016/j.jsv.2005.03.007.

Sak, H. S., Andrew / Beaufays, Françoise (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. INTERSPEECH 2014, 338-342.

Samanta, B., & Nararaj, C. (2009). Use of particle swarm optimization for machinery fault detection. Engineering Applications of Artificial Intelligence, 22(2), 308-316, doi:https://doi.org/10.1016/j.engappai.2008.07.006.

Schoen, R. R., Habetler, T. G., Kamran, F., & Bartfield, R. G. (1995). Motor bearing damage detection using stator current monitoring. IEEE Transactions on Industry Applications, 31(6), 1274-1279, doi:10.1109/28.475697.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W. K., & Woo, W.-c. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Vol. 15).

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM Neural Networks for Language Modeling.

Wang, Y., Liu, F., & Zha, A. (2019). Bearing Fault Diagnosis Based on a Hybrid Classifier Ensemble Approach and the Improved Dempster-Shafer Theory. Sensors (Basel), 19(9), doi:10.3390/s19092097.

Yao, H., Wu, F., ke, J., Tang, X., Jia, Y., Lu, S., et al. (2018). Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction.

Zhang, R., Peng, Z., Wu, L., Yao, B., & Guan, Y. (2017). Fault Diagnosis from Raw Sensor Data Using Deep Neural Networks Considering Temporal Coherence (Vol. 17).

Zhen Zuo, B. S., Wang Gang, Xiao Liu, Xingxing Wang, Bing Wang, Yushi Chen (2016). Learning Contextual Dependencies with Convolutional Hierarchical Recurrent Neural Networks. IEEE Transactions on Image Processing, 10.1109/TIP.2016.2548241.

Zhou, W., Habetler, T. G., & Harley, R. G. (2008). Bearing Fault Detection Via Stator Current Noise Cancellation and Statistical Control. IEEE Transactions on Industrial Electronics, 55(12), 4260-4269, doi:10.1109/TIE.2008.2005018.