# Vibration Anomaly Detection using Deep Neural Network and Convolutional Neural Network

Crina Deac, Gicu Călin Deac, Radu Constantin Parpală, Cicerone Laurentiu Popa, and Costel Emil Cotet

***Abstract*—Identifying the "health state" of the equipment is the domain of condition monitoring. The paper proposes a study of two models: DNN (Deep Neural Network) and CNN (Convolutional Neural Network) over an existent dataset provided by Case Western Reserve University for analyzing vibrations in fault diagnosis. After the model is trained on the windowed dataset using an optimal learning rate, minimizing the cost function, and is tested by computing the loss, accuracy and precision across the results, the weights are saved, and the models can be tested on other real data. The trained model recognizes raw time series data collected by micro electro-mechanical accelerometer sensors and detects anomalies based on former times series entries.**

***Index Terms*—DNN, CNN, anomaly detection, fault diagnosis, vibration analyses condition monitoring, industry 4.0.**

## I. Introduction

Any machine, whether it is a rotating machine (steam turbine, pump, compressor, etc.) or a non-rotating machine (heat exchanger, valve, etc.) will eventually touch a point of poor health. The point might not be that of an actual failure, but one at which the equipment signals that it is no longer in its optimal state. This indicates that maintenance is required. The most used way to perform condition monitoring is to verify each sensor measurement from the machine and to impose a minimum and a maximum value limit on it. If the acquired value is in the interval, then the machine is healthy, otherwise an alarm is triggered, and the machine is considered to be in a poor condition. This procedure is known to send a large number of false alarms or misses alarms when situation calls for it, wasting time and availability of the equipment, or leading to failures [1]. In this paper authors propose another diagnosis method that uses the same measurements type differently to better evaluate the real state of the machine. The proposed solutions reside in machine learning sphere based on statistical analysis for anomaly detection. There are several approaches: using multivariate statistics, performing a principal component analysis and then calculating the Mahalanobis distance to identify normal and anomalous data, some classifications algorithms including Support Vector Machine (SVM), using artificial neural network approach, like autoencoder networks, which compress the sensor reading to a lower dimensional representation, as PCA model, and captures the correlations and interactions between the various variables [2].

The present paper uses deep learning approach for the fault diagnosis. The method doesn't need a feature selection or decrease noise and depends only on the temporal coherence of the raw times series data supplied by the accelerometer. It is known that Deep Learning is able to reduce the dimensionality and learn characteristics from nonlinear data in the speech recognition, image classification and sentiment classification [3]. The proposed models are supervised learning models that recognizes the normal and fault data having a good accuracy in classification, learning directly from the raw time series and making use of the temporal coherence.

## II. Data Collection

In order to validate this methodology based on deep learning neural network for fault diagnosis, we have used data provided by Case Western Reserve University [8] (CWR) where motor bearing experiments were initiated in order to characterize the performance of a motor bearing condition assessment system. The CWR datasets were over the time the case of study also for other papers [11], [12].

The test stand used to acquire the bearing dataset consists of a 2hp motor, a torque transducer, dynamometer and control electronics. The test bearings support the motor shaft. Single point faults were introduced using electro-discharge machining to the test bearing (B1) having fault diameters of 0.007" and 0.021" at two different rotations: 1797 rpm, motor load (HP) 0 and 1772 rpm, motor load (HP) 1.

Outer raceway faults are stationary faults as the outer race is fixed, as a result the placement of the fault relative to the load zone of the bearing has a direct implication on the vibration response of the system [8]. The drive end bearing have the outer raceway faults located in the load zone (6 o'clock), at 3 o'clock (orthogonal to the load zone) and at 12 o'clock.

All data sets were collected using accelerometers attached to the motor housing with magnetic bases and placed at the 12 o'clock position at both the drive end (bearing B1) and fan end (bearing B2) for each rotational speed. Data was collected at 12,000 samples /second for drive end bearing experiments (B1) and fan end bearing experiments (B2).

The drive end and fan end were both considered for our training. Datasets where separately considered for two fault diameters 0.007" and 0.021" in two different load cases: motor Load 0 hp and motor Load 1 hp, with 5 types of faults:

Manuscript received February 20, 2020; revised June 23, 2020.

The authors are with the University "Politehnica" of Bucharest, Romania (e-mail: george.deac@impromedia.ro, crina.deac@impromedia.ro, radu.parpala@gmail.com, laur.popa79@gmail.com, costelemilcotet@gmail.com).

DOI: 10.7763/IJMO.2021.V11.772

inner race fault, ball fault and 3 outer race faults (centered – orthogonal – opposite) disponible in the dataset provided by Case Western Reserve.

For the first case - Load 0, the rotational speed of the engine was 1797 rotations per minute (RPM) and as the sampling rate was set at 12 000/s, it means 12000 data points recorded in a second, 1797 Rot/min equals 30 Rot/s, it implies we have 400 data points by one rotation. In the second case – Load 1, the rotational speed was 1772 rotations per minute (RPM) at the same sampling rate, it means 29.53Rot/s. The rotational speed was approximated for both cases at 30 per second.

In order to validate the model, in the experiment models were trained on one load and one diameter fault case, including both bearings B1 and B2, 6 files have been used for each bearing containing fault data and also the normal case, so that we had for each one:

1 file for normal behavior containing 483903 datapoints

for diameter fault case of 0.021":

1 file for inner race fault containing 121991 datapoints,

1 file for outer raceway centered fault (@6:00, position relative to load zone, load zone centered at 6:00) containing 121426 datapoints,

1 file for outer raceway opposite fault (@12:00) containing 121991 datapoints,

1 file for outer raceway orthogonal fault (@3:00) containing 121846datapoints,

1 file for roller defect containing 121410 datapoints

A total of 1012207 data points included in 6 files were used for B1 and the same total for B2, the sum of sets for those two bearings, included in 12 files, will make up the buffer size of our dataset. Each data point contains only one component (the acceleration on one axis). Data obtained from all these 12 files where used to obtain 12 timeseries that were processed in order to train, validate and test several models on both the drive end bearing B1 and the fan end bearing B2.

This means six types of data are collected in the merged dataset, sets that were labeled within 6 classes (label 0 for normal condition, 1 for inner race fault, 2 for outer race centered, 3 for outer race opposite, 4 for outer race orthogonal fault, 5 for roller defect).

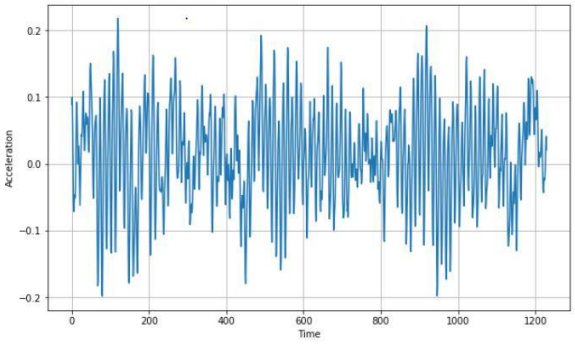



Fig. 1. Drive end bearing B1-normal behavior – Label 0.

Considering that the data size is 1012207 points x 2 (in the exemplification case) and that data can't be served as an input of a DNN, data should be divided to form samples. The rotation period consists of 400 data points. Considering this, data was divided by the size of the sampling period (sampling period is approx.400 points), so that each sequences of data containing 400 data points (window size) will be a features vectors for the model. [4].

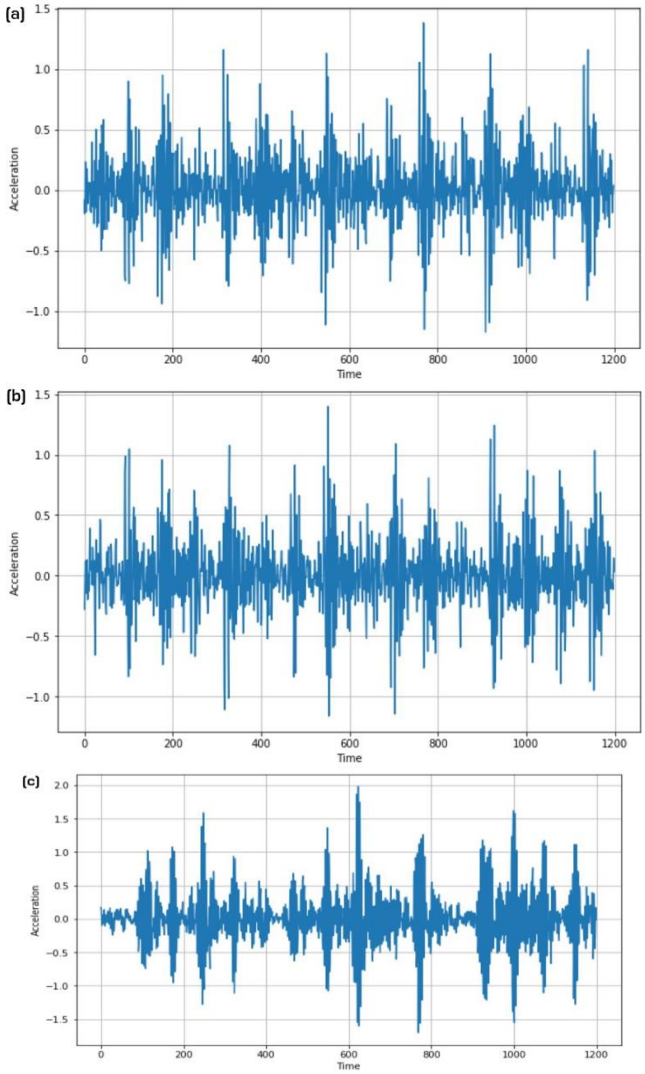
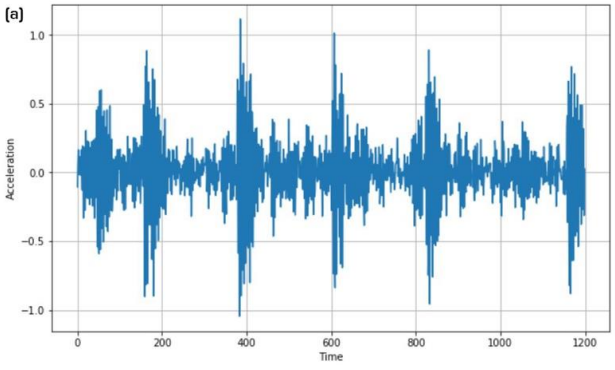



Fig. 2. Drive end bearing B1, inner raceway fault Label 1. (a) 0,007" Load 0, (b)0,007" Load 1, (c) 0,021" Load1.

During experiments better results were obtained when the window size was changed from 400 points, that means one full rotation, to 100 points representing a quart of rotation. As an example, by changing the segmentation [4], the accuracy of DNN model was improved from 82.75% to 94.56% for the experiments conducted on datasets in Load 0, with a fault diameter of 0.007". Also, better results for DNN model were obtained when switching from 4 Dense layers to 5 Dense layers and using a smaller batch size of 20 samples.



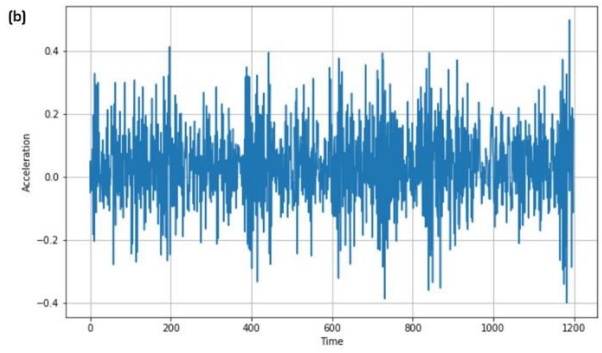



Fig. 3. Outer raceway opposite fault label 3.
(a) B1, 0.007" Load 1, (b) B2, 0.007" Load1.

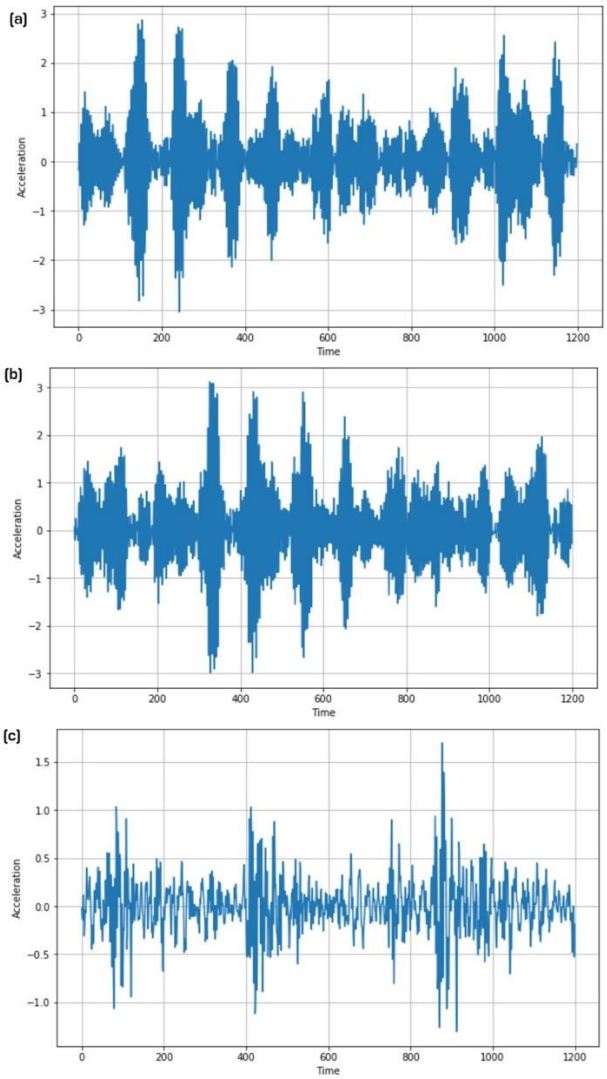
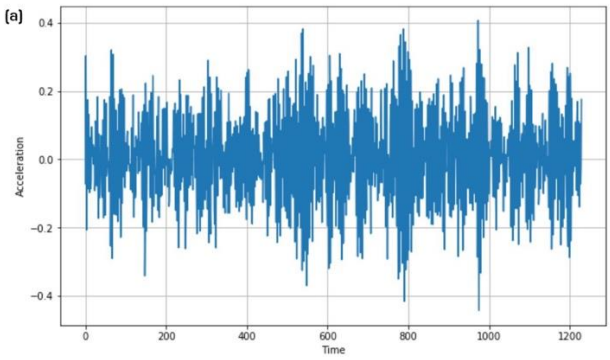



Fig. 4. Drive end bearing B1 outer raceway orthogonal fault label4.
(a)0.007" Load0, (b)0.007" Load1, (c)0.021"Load1.



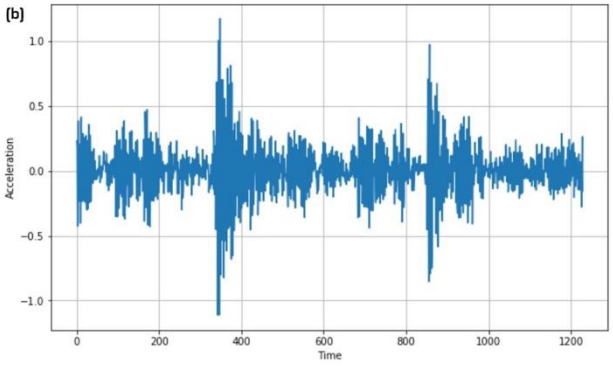




Fig. 5. Drive end bearing B1 roller fault Label 5.
(a) 0.007" Load 1 and (b) 0.021" Load 1.

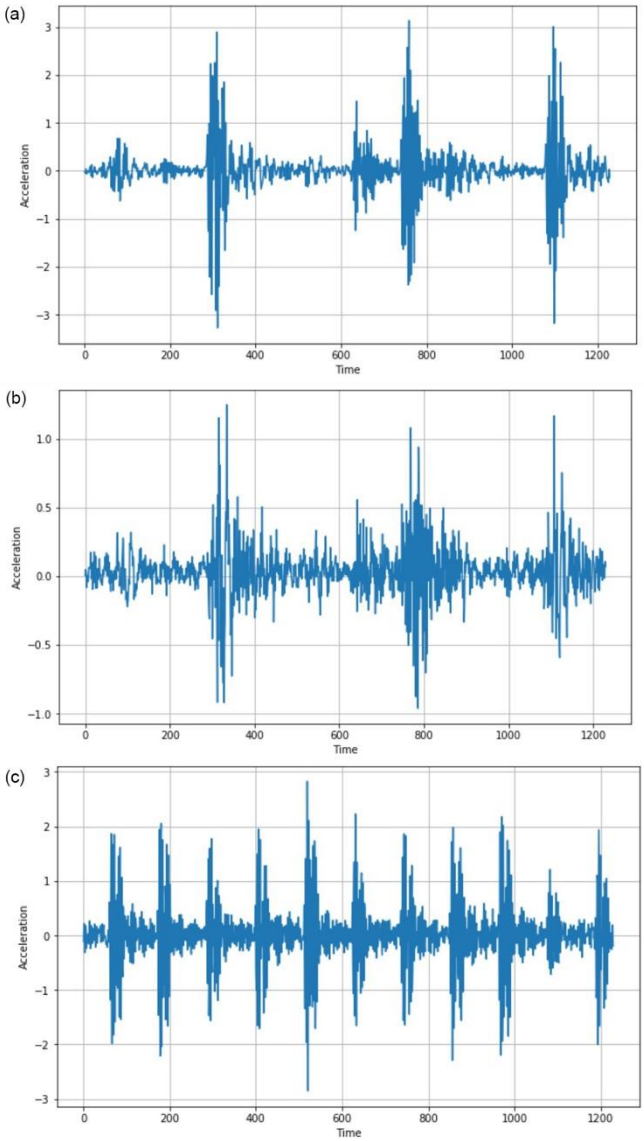



Fig.6. Outer raceway centered fault label 2.
(a) B1, 0.021" Load 1, (b) B2, 0.021" Load 1, (c) B1, 0.007" Load1.

A batch size of 20 samples was used to train the model to predict the 6 classes (0, 1, 2, 3, 4 or 5). The model will match a window of 100 features to a single label.

Thus, 1012207 data points for all 12 files will be fragmented in 10122 total samples for all 6 kind of data, which will feed the model in the case study.

Before this fragmentation, each of the 12 series was split once at 70% and once again at 85% in order to obtain 70% of

data forming the training set, 15 % of data forming the validation dataset and finally, the rest of 15 %, composing the test set. After splitting all sets, the first 70% from each time series were concatenate to compose the training dataset, then the next 15% to compose validation dataset, this was done in order to prevent the overfitting and to select the best trained neural network. The rest of 15 % from each timeseries was used to form the test dataset that will be used to compute the classification accuracy of the model. The sampling was automatically generated by using a windowed dataset function that uses flat_map to make sure the order of windowed datapoints, which represents tensors of 100 datapoints, stays the same, also it does a one hot encoding inside, for the labels, after that, shuffles the samples. The one hot encoding will convert the categorical value of Label 0 into a tensor like [1 0 0 0 0 0] and the categorical value of Label 5 into a sensor like [0 0 0 0 0 1].

TABLE I: LABELED DATASETS

| Data Type | No. Samples (B1+B2) | Label |
|---|---|---|
| Normal | 4839 × 2 | 0 |
| Defect in the Inner race | 1219 × 2 | 1 |
| Defect in the Outer race (centered) | 1214 × 2 | 2 |
| Defect in the Outer race (opposite) | 1219 × 2 | 3 |
| Defect in the Outer race (orthogonal) | 1218 × 2 | 4 |
| Defect in the Roller | 1214 × 2 | 5 |

## III. DNN MODEL (DEEP NEURAL NETWORK)

A deep neural network structure model is used to learn characteristics based on temporal coherence from raw series data and make diagnosis. Deep Neural Network can learn useful features from data adaptively without expertise in specific fields. It has stacked layers of units, connected layer by layer, but there is no connection among the units in the same layer. The deep learning network has an input layer, an output layer and also several hidden layers between the input and output layer. The number of input units are set according the dimensionality of the input data (the window size) and the number of output units are set according the target data (in our case to 6, because we have 6 labels). In each layer unit there is applied an activation function $\sigma(z)$ to a linear combination of former activations and added a bias:

$$a_j^m = \sigma(z_j^m),$$

where $m$ is the layer number, $j$ the neural unit,

$z_j^m = \sum \omega_{ij} a_i^{m-1} + b_j^m$ the linear combination of former activations in layer $m$-1

where:

$\sigma(z) = \max(0, z)$ is the ReLU activation function and

$\sigma(z) = 1/(e\text{-}z + 1)$ is the sigmoid activation function used in the final layer. $\omega$ and $b$ are parameters for the model, first they are randomly initialized and then they are optimized during the epochs by the model. All units in the next layer are connected to every units from the formal layer.

TensorFlow 2.0 and NumPy libraries were used for scientific computing in Python, Matplotlib Pyplot for plotting data, Os for parsing the files from the folder, Pandas for reading csv files and composing the data set in dataFrame, sklearn for metrics and Keras for defining the DNN deep learning model.

The model was defined on a 5 layers neural network: first layer having 400 neurons, the input shape being the size of our window of 100 points, the next three layers having 100 units, all layers with rectified linear activation function ReLU, and the last layer having 6 nodes, one unit for each class type and the Sigmoid activation function in order to predict the probability for each class.

The model was defined with the binary cross entropy error loss function (log loss). Cross-entropy quantifies the difference between two probabilities distribution. The model predicts a model distribution of $\{p,1\text{-}p\}$ (binary distribution) for each of the classes. Binary cross entropy will be used to compare these with the true distributions $\{y,1\text{-}y\}$ for each class and sum up their results:

$$\boldsymbol{C}(y_i, y_{i\_predict}) = \text{Binary Cross Entropy}$$
$$= -\sum_i^m (y_i \log(y_{i\_predict}) + (1-y_i)\log(1-y_{i\_predict})),$$

where $y$ is the true value and $y_{predict}$ is the predicted value $p(y_i)$, the output of the model.

The scope is to minimize the cost function by using gradient descent method. The model was trained using the "SGD" Stochastic Gradient Descent optimizer, for the first 100 epochs in order to choose the best learning rate, and then for 400 epochs, to minimize the score, by computing partial derivatives of the cost function and updating the parameters and learning rate. $\frac{\partial C}{\partial \omega}, \frac{\partial C}{\partial b}$.

A callback function was used to compute the best learning rate, the value of the learning rate is afterwards adjusted each epoch. The learning rate was plotted versus loss values obtained in each epoch and that value "lr_schedule", which minimizes the most the loss, was pick up.

```
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
  lambda epoch: 1e-8 * 10**(epoch / 20)
)
```

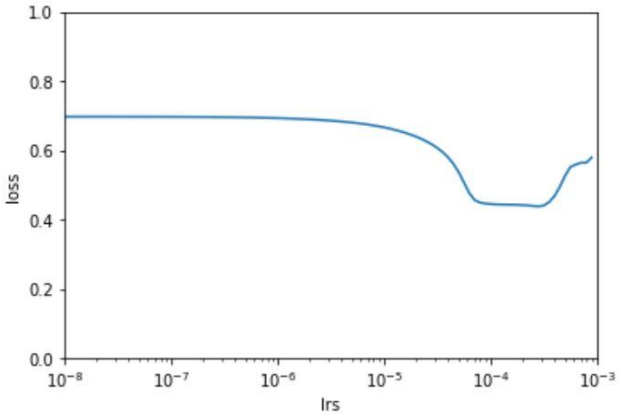



Fig. 7. Loss value function vs. learning rate variation for experiment Load 0+1, all fault diameters, B1+B2

For the experiment showed in Fig. 7 a learning rate of 5e-4, that minimizes the loss, was chosen.

Then, the iteration process was continued over the number of chosen epochs until model fitted. (for different experiments a different value was chose 400 or 200 depending on the case).

```
model = tf.keras.models.Sequential([
```

```
    tf.keras.layers.Dense(400, input_shape=[window_size], activation="relu"),
    tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(6, activation="sigmoid")
])
  Optimizer=tf.keras.optimizers.SGD( lr=5e-4,  momentum = 0.9)
model.compile (loss = "binary_crossentropy", optimizer =
  optimizer, metrics=['accuracy'])
  history = model.fit (dataset, epochs=400, validation_data = (dataset_valid), verbose=1)
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_5 (Dense) | (None, 400) | 40400 |
| dense_6 (Dense) | (None, 100) | 40100 |
| dense_7 (Dense) | (None, 100) | 10100 |
| dense_8 (Dense) | (None, 100) | 10100 |
| dense_9 (Dense) | (None, 6) | 606 |

Total params: 101,306
Trainable params: 101,306
Non-trainable params: 0

Fig. 8. DNN model summary.

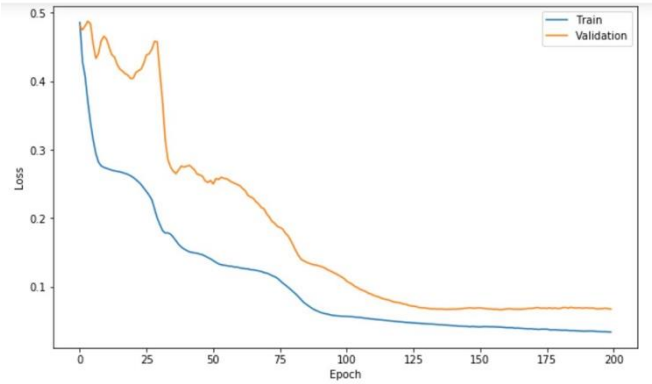



Fig. 9. Loss variation during 200 epochs over B1+B2, Load 0 +Load 1, all fault diameters.

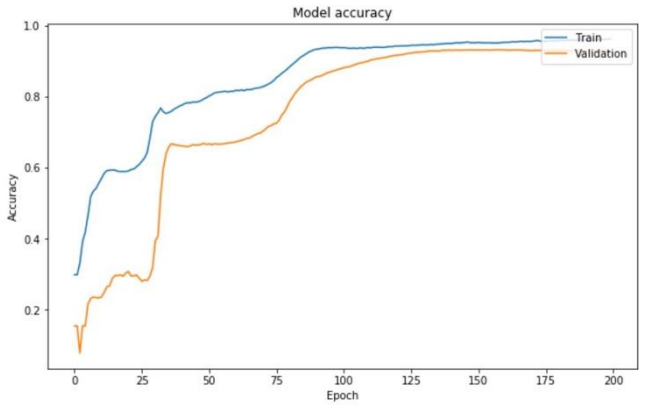



Fig. 10. Accuracy variation during 200 epochs, over B1+B2, Load 0 +Load 1, all fault diameters.

After the first stage, training the model, the process was tested for fault recognition process and fault recognition using a test dataset unknown to our model.

The confusion matrix in the test set is showing the predicted values on the columns for each label (0,1,2,3,4,5) and true positives and true negatives (the normal - Label 0) on the first diagonal.

Labels 4 and 2 are most accurate recognized by the model after the normal (Label 0) which are almost 100% recognized, in majority of the experiments. On the first diagonal, as it can be seen in the figure, they are colored lighter and normal examples are almost white colored.

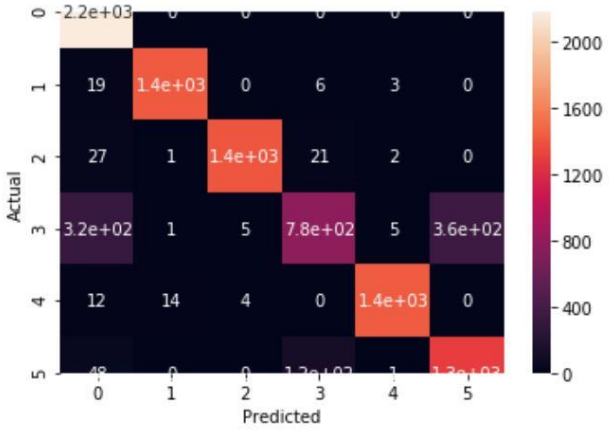



Fig. 11. Confusion matrix on the test dataset B1+B2, Load 0 + Load 1, all fault diameters.

## IV. CNN Model (Convolutional Neural Networks)

An optimized CNN model was after that tested in order to get better predictions. The model was defined as a sequential Keras model for simplicity [9]. In the first stage convolutions with LSTM [10] where combined but that didn't conduct to better results.

Finally, a second model was defined having: two convolutional 1D layers, 12 filters with a kernel size of 5, a max-pooling layer, another sequence of two convolutional 1D layers, a global average pooling layer and finally a Dense layer for output. The final layer, activated by sigmoid function, containing 6 units because the output must fit the number of labels.

The 1D convolutional layers will try to learn 12 filters, it will take a five number window and multiply out the values in that window by the filter values, in this way some features being highlighted [5]. Then the max pooling layer reduces the learned features to 1/3 of their size, choosing from three consecutive values the maximum one, consolidating the features to only the most essential elements [7]. The next set of two convolutional 1D layers give the model a good chance of learning features from the input data [10] and then, again, global pooling layer aggressively summarize the presence of a feature and is used as a transition from feature maps to the output prediction [6]. The output of the global average pooling layer is a single value that summarizes the presence of the feature in the single feature map.

The model was compiled with the binary cross entropy error loss function (log loss), as in the case of DNN model, and trained using the "SGD" Stochastic Gradient Descend optimizer, first for 100 epochs in order to select the best learning rate. Then the model was trained for another 200 epochs, to minimize the score by computing partial derivatives of the cost function and updating the parameters and learning rate.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, None, 1)           0
_________________________________________________________________
conv1d (Conv1D)              (None, None, 12)          72
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 12)          732
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, None, 12)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, None, 12)          732
_________________________________________________________________
conv1d_3 (Conv1D)            (None, None, 12)          732
_________________________________________________________________
global_average_pooling1d (Gl (None, 12)                0
_________________________________________________________________
dense (Dense)                (None, 6)                 78
=================================================================
Total params: 2,346
Trainable params: 2,346
Non-trainable params: 0
```

Fig. 12. CNN model summary.

## V. Testing DNN And CNN Models On Various Data Sets

First, the two models were trained only for the drive end bearing B1 on normal and faulty behavior (fault of 0.007") at a constant load and speed. The trained model was tested for the same bearing but under different conditions: 1. different load with the same fault diameter, 2. the same load and speed but with different fault diameter. This was done in order to evaluate model performances. Good results were obtained on the dataset where the DNN model was used using the following datasets: Load 1, B1 drive end bearing dataset with a fault diameter of 0.007". The accuracy was 100% on the train set, 90,05% on the validation set, 91,56% on the test dataset.

Classification report on test dataset (Table II) shows in recall column the percent of detection for each label. One can see that only the fault in the roller (Label 5) has a poor detection accuracy of only 45% (recall 0.45). This could be explained by the fact that this fault doesn't occur on each rotation as the fault can be in a position that it won't affect the bearing behavior (between the inner and the outer race). In this case the training data set for the ball fault will look alike with the one recorded for normal behavior.

TABLE II: Classification Report on Test Dataset B1, Load 1, Fault Diameter 0.007", DNN Model

| Label | Precision | Recall | F1 score |
|---|---|---|---|
| Label 0 | 1 | 1 | 1 |
| Label 1 | 0.99 | 1 | 1 |
| Label 2 | 0.97 | 0.97 | 0.97 |
| Label 3 | 0.59 | 0.81 | 0.68 |
| Label 4 | 1 | 1 | 1 |
| Label 5 | 0.74 | 0.45 | 0.55 |

TABLE III: Results for Model Trained on B1, Load 1, Fault Diameter 0.007"

| **Model DNN** | **Load** | **Diameter fault** | **Accuracy on test dataset** |
|---|---|---|---|
| Model was trained only on B1- normal and 5 faults types data | Load 1 | 0.007" | 91,56% |
| Tested only on faults data (B1) | Load 1 | 0.007" | 96,73% |
| Tested only on faults data (B1) | Load 1 | 0.021" | 33,43% |
| Tested only on faults data (B1) | Load 0 | 0.007" | 83,10% |
| Tested only on faults data (B1) | Load 0 | 0.021" | 29,87% |
| Tested on normal examples (B1) | Load 1 | - | 100% |
| Tested on normal examples (B1) | Load 0 | - | 100% |

When tested on Load 0 the model that was trained on Load 1 with a fault diameter of 0.007", a good accuracy of 83.10% was also obtained, but the same model recorded a poor accuracy, of less than 40%, on datasets having a different fault diameter (0.021") (Table III):

Good results were obtained for both load cases using the normal dataset for B1 unlike tests conducted on datasets with a fault diameter of 0.021" (see Table IV).

TABLE IV: Accuracy on Test Dataset Having 0.021" Fault Diameter

| Load | Label1 | Label2 | Label3 | Label4 | Label5 |
|---|---|---|---|---|---|
| Load 1 | 20.3% | 29,04% | 37,90% | 3% | 76.66% |
| Load 0 | 17,8% | 27,70% | 28,90% | 4% | 70,30% |

TABLE V: Results for DNN Model Trained on B1+B2, Load 1, Fault Diameter 0.007"

| **Model DNN, LR=5e-4, 500 epochs** | **Load** | **Diam fault** | **Accuracy on test dataset** | **Precision** |
|---|---|---|---|---|
| Model was trained on normal and 5 faults types (B1 and B2) | Load 1 | 0.007 | 90,40% | 90,19% |
| Tested on B1 dataset, only faults (5) | Load 1 | 0,007 | 96,37% | 96,30% |
| Tested on B2 dataset only faults (5) | Load 1 | 0.007 | 98,38% | 98,34% |
| Tested on B1 dataset, only faults (5) | Load 1 | 0.021 | 30,28% | 28,66% |
| Tested on B2 dataset only faults (5) | Load 1 | 0.021 | 29,04% | 29,43 |
| Tested on all normal (B1 + B2) | Load 0 | | 99,94% | 99,96% |

In Table IV one can notice that only for the defect in the roller (Label 5) the model achieved a satisfactory detection rate and also for the outer race orthogonal fault (Label 4) the model recorded the weakest detection rate.

For the CNN model, the overall accuracy obtained on test dataset was 89,79%, which is less than the one from the DNN model. The main reason for this poor result was that for Load 1 and a fault diameter of 0.007" the Label 3 wasn't at all detected.

The next experiment was to train the models using both data, from B1 and B2, for one Load type (Load 1 or Load 0) and to see if the model can find a pattern that match both bearings behaviors and if the predicted results are improved. The unidimensional data sets, provided by accelerometers installed on B1 and B2 were concatenate, obtaining a unidimensional series. The results show that the model was able to predict very well on both B1 and B2, for the same faults diameters and for the same load (Table 5 and Table 19). For data provided by B1 and B2, in Load 1 case, 0.007"

diameter fault, the following results were obtained: 100% accuracy on training dataset, 95% on validation dataset and 90,4% on test dataset accuracy (Table V):

As one can see from Table V training the model on a set including normal and faulty behavior for one load and one diameter and testing on another set (another load) with normal behavior yield good results over 99% accuracy. One problem arises when testing on faulty data this model predicted faulty behavior with a good accuracy but was unable to accurately identify the fault type, results indicated an accuracy of less than 50% for fault identifications.

For condition monitoring it is important to predict failures as early as possible in order to prevent machines failure and this model is able to report a failure in its early stage. In order to improve further fault identification a better approach will be to group similar defects under one single label, as an example for the actual dataset there are 3 labels for the outer race faults (centered, orthogonal and opposite) those faults generating a similar acceleration profile.

Classification report generated on test dataset (B1+B2), Load 1, 0.007" diameter fault, shows that Label 5 (defect in the roller) was again predicted with a precision of less than 50% (recall 0.41):

TABLE VI: CLASSIFICATION REPORT ON TEST DATASET, B1+B2, LOAD 1, FAULT DIAMETER 0.007", DNN MODEL

| Label | Precision | Recall | F1 score |
|---|---|---|---|
| Label 0 | 0.98 | 1 | 0.99 |
| Label 1 | 1 | 0.99 | 0.99 |
| Label 2 | 0.96 | 0.97 | 0.97 |
| Label 3 | 0.55 | 0.70 | 0.62 |
| Label 4 | 0.99 | 0.98 | 0.99 |
| Label 5 | 0.62 | 0.41 | 0.50 |

When DNN model was trained on data having 0.007" diameter fault, Load1, tests on datasets having fault diameters of 0.021", at Load 1, didn't obtain more than 30.28% overall accuracy on all faults types (Table VI).

When training the DNN model on datasets having the diameter fault 0.021", at Load 1 (Table VIX), tests on dataset having 0.007" fault diameter obtained 36,35% accuracy and on the entire dataset where the model was trained the following results were obtained: 100% accuracy on training dataset, 83,53% on validation dataset, 87,07% on test dataset and a classification report like in Table VIII. In this case the detection gets better for Label 5 defect in the roller (87%) and worse for Label 2 (61%).

TABLE VII: CLASSIFICATION REPORT ON TEST DATASET, B1+B2, LOAD 1, FAULT DIAMETER 0.021", DNN MODEL

| Label | Precision | Recall | F1 score |
|---|---|---|---|
| Label 0 | 0.96 | 1 | 0.98 |
| Label 1 | 0.96 | 0.76 | 0.85 |
| Label 2 | 0.61 | 0.67 | 0.64 |
| Label 3 | 0.64 | 0.63 | 0.63 |
| Label 4 | 0.91 | 0.87 | 0.89 |
| Label 5 | 0.86 | 0.87 | 0.86 |

TABLE VIII: RESULTS FOR MODEL TRAINED ON B1+B2, LOAD 1, FAULT DIAMETER 0.021"

| **Model DNN, LR=1e-3, 500 epochs** | **Load** | **Diam fault** | **Accuracy on test dataset** | **Precision** |
|---|---|---|---|---|
| Model was trained on normal and 5 types faults data (B1 and B2) | Load 1 | 0.021 | 87,07% | 87,62% |
| Tested only on all faults data 5 types (B1) | Load 1 | 0.021 | 92,80% | 92,90% |
| Tested only on all faults data-5 types (B2) | Load 1 | 0.021 | 91,89% | 92,35% |
| Tested only on all faults data- 5 types (B1) | Load 1 | 0.007 | 35,63% | 36,20% |
| Tested only on all faults data- 5 types (B2) | Load 1 | 0.007 | 35,20% | 35,20% |
| Tested on all normal (B1 + B2) | Load 1 | | 99,90% | 99,98% |
| Tested on all normal (B1 + B2) | Load 0 | | 99,82% | 99,84% |
| Tested only on all faults data 5 types (B1) | Load 0 | 0.007 | 31,61% | 31,94% |

Results from the DNN model have a poor accuracy on diameter faults where the model wasn't trained.

Testing the DNN model trained on 0.021" diameter faults on data having 0,007" diameter fault, at the same Load 1, an average accuracy of 35,63% was obtained, as one can see in the Table VIII. In this case, the confusion matrix at evaluation the model only on data having faults 0.007" diameter shows in Fig. 13, on the rows, the existent real data from the evaluation test dataset (only faults: Label 1 to Label 5) and on the columns shows the predictions (from Label 0 to Label 5). It can be seen that a small part from the faults were predicted as normal (see column Label 0): 97 samples actually Label 1, 32 samples actually Label 2, 21 samples actually Label 3, 5 samples actually Label 4 and finally 13 samples actually being Label 5 were all allocated to normal. On the first diagonal of the matrix (columns 1-5) one can see the correct predictions, all other faults are incorrect allocated to other faults types. Reading from row 1 which comprises only real faults belonging to Label 1, it can be noticed that 97 were allocated to normal, 63 correct predicted, 579 allocated to fault Label 2, 21 predicted as fault Label 3 and 459 allocated to fault Label 4. Generally, only 2% were predicted as normal from 100% data having faults, 36% were correct identified and the rest of 62% were allocated to other faults types.

| **Predicted** | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| **Actual** | | | | | | |
| 1 | 97 | 63 | 579 | 21 | 459 | 0 |
| 2 | 32 | 49 | 232 | 911 | 0 | 0 |
| 3 | 21 | 17 | 69 | 888 | 0 | 224 |
| 4 | 5 | 1168 | 0 | 5 | 40 | 0 |
| 5 | 13 | 7 | 4 | 308 | 0 | 882 |

Fig. 13. Confusion matrix on test dataset having 0.007" diameter fault at Load1, only faults data, DNN model.

Testing the DNN model trained on data at Load 1 (Table VIII) over normal samples but other load (Load 0), has been again obtain a very good accuracy of 99,82%, normal examples being very well detected, even when changing the Load.

Thus, in the next experiment datasets were unified from both diameters' faults: 0.007" and 0.021", from both bearings B1 and B2 and train the two models for a specific Load type. In this case both DNN and CNN models were improved by modifying the sequential layers, by tuning the learning rate and by adjusting the window size of the sampling. DNN model gave an accuracy of 100% on the training dataset, 89,26% on validation data set and 92,20% on test dataset while CNN model obtained an accuracy of 99,42% on training dataset, a better accuracy of 97,82% on validation dataset and 96,17% on test dataset in Load 1 case, having both diameters faults inside data collection.

TABLE IX: RESULTS FOR MODEL DNN TRAINED ON B1+B2, LOAD 1, FAULT DIAMETERS 0.021" + 0.007"

| **Model DNN, LR=1e-3, 500 epochs** | **Load** | **Diam. fault** | **Accuracy on test dataset** | **Precision** |
|---|---|---|---|---|
| Model was trained on 70% from normal and 5 faults types data, (B1 and B2) | Load 1 | 0.021" + 0.007" | 92,20% | 92,28% |
| Tested on all B1 + B2, normal data | Load1 | | 99,98% | 99,98% |
| Tested on all B1 + B2, on all fault data | Load 1 | 0.021" + 0.007" | 65,09% | 65,02% |
| Tested on all B1 + B2, Label 1 | Load1 | 0.007" | 99,79% | 99,71% |
| Tested on all B1 + B2, Label 2 | Load1 | 0.007" | 98,61% | 98,61% |
| Tested on all B1 + B2, Label 3 | Load1 | 0.007" | 95,24% | 95,37% |
| Tested on all B1 + B2, Label 4 | Load1 | 0.007" | 99,75% | 99,67% |
| Tested on all B1 + B2, Label 5 | Load1 | 0.007" | 95,14% | 95,45% |
| Tested on all B1 + B2, Label 1 | Load1 | 0.021" | 0,4% | 0,3% |
| Tested on all B1 + B2, Label 2 | Load1 | 0.021" | 8% | 7% |
| Tested on all B1 + B2, Label 3 | Load1 | 0.021" | 48,20% | 49,11% |
| Tested on all B1 + B2, Label 4 | Load1 | 0.021" | 24,82% | 21,78% |
| Tested on all B1 + B2, Label 5 | Load1 | 0.021" | 80,73% | 81,59% |
| Tested on all B1 + B2 normal data | Load 0 | | 99,90% | 99,94% |
| Tested on all B1 + B2, Label 1 | Load 0 | 0.007" | 93,85% | 94,63 |
| Tested on all B1 + B2, Label 2 | Load 0 | 0.007" | 91,55% | 93,30% |
| Tested on all B1 + B2, Label 3 | Load 0 | 0.007" | 88,09% | 88% |
| Tested on all B1 + B2, Label 4 | Load 0 | 0.007" | 91,73% | 91,97% |
| Tested on all B1 + B2, Label 5 | Load 0 | 0.007" | 17,76% | 17,82% |
| Tested on all B1 + B2, Label 1 | Load 0 | 0.021" | 1% | 0.8% |
| Tested on all B1 + B2, Label 2 | Load 0 | 0.021" | 6% | 6% |
| Tested on all B1 + B2, Label 3 | Load 0 | 0.021" | 40,76% | 40,28% |
| Tested on all B1 + B2, Label 4 | Load 0 | 0.021" | 28,10% | 25,52% |
| Tested on all B1 + B2, Label 5 | Load 0 | 0.021" | 40,94% | 41,04% |

TABLE X: RESULTS FOR MODEL CNN TRAINED ON B1+B2, LOAD 1, FAULT DIAMETERS 0.021" + 0.007"

| **Model CNN LR=6e-4, 500 epochs** | **Load** | **Diam. fault** | **Accuracy on test dataset** | **Precision** |
|---|---|---|---|---|
| Model was trained on 70% from normal and 5 faults types data, (B1 and B2) | Load 1 | 0.021" + 0.005" | 96,17% | 96,19% |
| Tested on all B1 + B2, normal data | Load 1 | | 99,99% | 99,98% |
| Tested on all B1 + B2, on all fault data | Load 1 | 0.021" + 0.005" | 61,98% | 67,65 |
| Tested on all B1 + B2, Label 1 | Load1 | 0.007" | 99,47% | 99,02% |
| Tested on all B1 + B2, Label 2 | Load1 | 0.007" | 99,22% | 97,86% |
| Tested on all B1 + B2, Label 3 | Load1 | 0.007" | 87,20% | 87,90% |
| Tested on all B1 + B2, Label 4 | Load1 | 0.007" | 99,88% | 99,14% |
| Tested on all B1 + B2, Label 5 | Load1 | 0.007" | 99,92% | 99,84% |
| Tested on all B1 + B2, Label 1 | Load1 | 0.021" | 4% | 0.1% |
| Tested on all B1 + B2, Label 2 | Load1 | 0.021" | 18% | 18% |
| Tested on all B1 + B2, Label 3 | Load1 | 0.021" | 22,30% | 18,71% |
| Tested on all B1 + B2, Label 4 | Load1 | 0.021" | 12,88% | 15,06% |
| Tested on all B1 + B2, Label 5 | Load1 | 0.021" | 76,17% | 77,59% |
| Tested on all B1 + B2, Label 1 | Load 0 | 0.007" | 98,10% | 96,41% |
| Tested on all B1 + B2, Label 2 | Load 0 | 0.007" | 96,72% | 96,10% |
| Tested on all B1 + B2, Label 3 | Load 0 | 0.007" | 87,36% | 88,01% |
| Tested on all B1 + B2, Label 4 | Load 0 | 0.007" | 99,63% | 98,97% |
| Tested on all B1 + B2, Label 5 | Load 0 | 0.007" | 93,43% | 93,29% |
| Tested on all B1 + B2, Label 1 | Load 0 | 0.021" | 5% | 3% |
| Tested on all B1 + B2, Label 2 | Load 0 | 0.021" | 16,67% | 16,54% |
| Tested on all B1 + B2, Label 3 | Load 0 | 0.021" | 29,02% | 29,96% |
| Tested on all B1 + B2, Label 4 | Load 0 | 0.021" | 8% | 10.58% |

| | | | | |
|---|---|---|---|---|
| Tested on all B1 + B2, Label 5 | Load 0 | 0.021" | 59,43% | 60,58% |

Each one of the models obtained good accuracy for normal data, almost 100%. Experimenting in Load 1 case, all faults were well detected on test dataset (DNN gave an accuracy of 92,20% - Table IX and CNN 96,17% - Table X).

In the case of the CNN model, excepts fault Label 3, where 127 samples were false detected as normal, 19 were wrongly associated to Label 4 and 105 samples wrongly predicted as having fault in the roller, all the other labels were correctly associated and true predictions for each labels are seen on the main diagonal of the matrix. (Fig. 14)

| Predicted | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Actual | | | | | | |
| 0 | 1450 | 0 | 0 | 0 | 0 | 0 |
| 1 | 7 | 727 | 0 | 0 | 2 | 0 |
| 2 | 15 | 0 | 720 | 1 | 4 | 0 |
| 3 | 127 | 1 | 3 | 467 | 19 | 105 |
| 4 | 0 | 5 | 0 | 0 | 721 | 0 |
| 5 | 3 | 0 | 0 | 0 | 0 | 719 |

Fig. 14. Confusion matrix on test dataset, CNN model, Load 1.

Classification report (Fig. 15) shows that for the Label 3 the recall is of only 61% meaning that only 61% from the examples in class 3 were correct predicted by CNN model. All other classes are well detected on test dataset which equals only 15% from entire dataset.

| Label | Precision | Recall | F1 score |
|---|---|---|---|
| Label 0 | 0.90 | 1 | 0.95 |
| Label 1 | 0.99 | 0.99 | 0.99 |
| Label 2 | 1 | 0.96 | 0.98 |
| Label 3 | 1 | 0.61 | 0.76 |
| Label 4 | 0.96 | 0.99 | 0.97 |
| Label 5 | 0.85 | 1 | 0.92 |

Fig. 15. Classification report on test dataset – CNN model trained on Load 1.

Evaluating the models only on faults on the entire dataset (for both B1+B2), both models obtained good accuracy in case of diameter fault of 0.007" but didn't obtained more than 48% accuracy for Labels 1,2,3 and 4 for diameter size of 0.021", except Label 5, the fault in the roller, that was better detected: accuracy 80.73% in case of DNN (Table IX), and accuracy of 99,92% in case of CNN (Table X).

Evaluations were also done by changing the Load: testing on Load 0 with data sets unknown to our model, similar results were obtained with tests on Load 1 for the faults labels (1,2,3,4) (Table IX and Table X).

For Load 0, it can be seen that CNN model provides the best fault detection in the roller, having an accuracy of 93,43% for 0.007" fault diameter and 59,43% for 0.021".

By combining all datasets from Load 0 and Load 1, with both B1 and B2 bearing data and both fault diameters: 0.007" and 0.021", the models slightly improved the detection for fault diameter of 0.021". When evaluating the CNN model trained on all diameters and all loads, an increased detection of around 10% for each label was obtained.

## VI. Conclusion

Using the two models, 5 types of bearings faults and normal behavior were predicted. All 5 types of faults used for training the models contained two different diameters size one of 0.007" and other of 0.021", this affecting the results, because the same type of defect was equally labeled both in the case of the diameter of 0.007 and in the case of the diameter of 0.021.

The shape of the signal in those two cases is different as one can see in Fig.4 (case b-0.007" and c-0.021" for Label 4), in Fig.5(case a-0.007"and b-0.021" for Label 5) or in Fig.6 (case a-0.021" and c-0.007" for Label 2) thus identifying a common pattern could rise some problems. In Fig.5 one could notice that the fault is better highlighted as the maximum amplitude generated by the fault rise over. The next approach could consist in defining, for the same fault, different labels based on fault diameter or training on sets recorder for faults with a more severe impact on overall behavior.

The presented models can diagnose the defects in incipient phase (for 0.007" fault diameter) with an accuracy of more than 95% and an accuracy of 100% for the normal behavior.

For the next phase of our research we intend to build a slightly different testing stand. The models trained on CWR datasets will be implemented on a monitoring a device for vibrations detection and diagnose. For convenience and portability, in the data collection and monitoring phase a Raspberry Pi 4 (quad-core arm cortex A72 @1.5Ghz) running Raspbian buster (a custom distro of ubuntu) could be proposed. The accelerometer communicates with the controller via an I2C interface configured on the GPIO. The sensor run at the maximum capable sampling rate of 12kHz providing acceleration data in the time domain on the three cartesian axis. The Python script on the device records the accelerations, writes the row data from the sensor in a csv text file containing 12000 data recorded in a second that will be given to the model to predict. Using the models trained on data provided by Case Western Reserve University we intend to detect faults in the bearings using the proposed detector.

## Acknowledgment

This work has been funded by the European Social Fund from the Sectoral Operational Program Human Capital 2014-2020, through the Financial Agreement with the title "Scholarships for entrepreneurial education among doctoral students and postdoctoral researchers (Be Antreprenor!)", Contract no. 51680/09.07.2019 - SMIS code: 124539

## Conflict of Interest

The authors declare no conflict of interest.

## Author Contributions

All authors had equal contributions in: the literature review, analyzing the data, writing the paper and approving the final version.

## References

[1] V. Flovik. How to use machine learning for anomaly detection and condition monitoring. [Online]. Available: http:// towardsdatascience.com.

[2] V. Flovik. Machine learning for anomaly detection and condition monitoring. [Online]. Available: http://towardsdatascience.com.
[3] L. Moroney, "Introduction to tensorflow for artificial intelligence, machine learning and deep learning," course 1 of 4 in the "TensorFlow in Practice Specialization," Deeplearning Ai.
[4] L. Moroney, "Sequences, time series and prediction, course 4 of 4 in the "tensorflow in practice specialization," Deeplearning Ai.
[5] L. Moroney, "Convolutional neural networks in tensorflow course 3 of 4 in the "tensorflow in practice specialization," Deeplearning Ai.
[6] Nils Ackerman, *Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences*.
[7] J. Brownlee. A gentle introduction to pooling layers for convolutional neural networks. [Online]. Available: https://machinelearningmastery.com/
[8] Bearings Vibration Data, Case Western Reserve University. [Online]. Available: csegroups.case.edu/bearingdatacenter/pages/12k-drive-end-bearing-fault-data
[9] Dive into deep learning. [Online]. Available: https://www. d2l.ai
[10] J. Brownlee, "Deep learning for time series forecasting," *Predict the Future with MLPs, CNNs and LSTMs in Python*, pp. 124-135, 2020.
[11] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler, "Deep learning algorithms for bearing fault diagnostics – A comprehensive review," *Mechanical Systems and Signal Processing*, 2020.
[12] R. Zhang. Z. Peng, L. F. Wu, B. B. Yao, and Y. Guan, "Fault diagnosis from raw sensor data using deep neural networks considering Temporal coherence," *MDPI Journals*, 2017.



**Crina-Narcisa Deac (Georgescu)** was born in Baia Mare in 1973, She is a PhD student at the University "Politehnica" of Bucharest, the master's degree in training techniques in the virtual environment at the University "Politehnica" of Bucharest in 2016, she is a mathematician in the University of North Baia Mare in 1997. She is an entrepreneur working at Impro-Media SRL (https://impromedia.eu) as cofounder and CEO. She has published some articles and books in VR, AR and predictive maintenance topics.

**Gicu-Calin Deac** was born in Baia Mare in 1970, he is a PhD student at the University "Politehnica" of Bucharest. He got the master's degree in training techniques in the Virtual Environment at University "Politehnica" of Bucharest in 2016, He is a dipl. Engineer, University of North Baia Mare in 1995. He is an entrepreneur working from at Impro-Media SRL (https://impromedia.eu) as cofounder and CTO. He has published some articles and books in VR, AR and IIoT topics.

**Constantin Radu Parpala** is a lecturer at the University Politehnica of Bucharest, Faculty of Industrial Engineering and Robotics, with a PhD in industrial engineering. He has published over 40 papers in scientific journals and conference proceedings. He participated as a researcher in more than 10 research projects. His main fields of interest include but are not limited to IoT, IIoT, manufactuing processes, design, FEM, industry 4.0, databases.

**Cicerone Laurentiu Popa** is an associate professor at the University Politehnica of Bucharest, Faculty of Industrial Engineering and Robotics, with a PhD in industrial engineering. He has published over 50 papers in scientific journals and conference proceedings. He was a project manager in the project *Selective Waste Collection Integrated System for a Smart City* – SMARTCOLLECT (2016-2018) and participated as a researcher in over 12 research projects. His research is industrial engineering, waste management, material flow management, smart cities, Industry 4.0.

**Costel Emil Cotet** is a professor at the University Politehnica of Bucharest, Faculty of Industrial Engineering and Robotics, with a PhD in industrial engineering. He has published over 50 papers in scientific journals and conference proceedings. He was project manager in three projects and participated as a researcher in over 40 research projects. His research is manufacturing architectures, virtual enterprises, industrial engineering, waste management, material flow management, smart cities, industry 4.0.