Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 1682874, 8 pages
https://doi.org/10.1155/2022/1682874

Research Article

Fault Diagnosis Using Data Fusion with Ensemble Deep Learning Technique in IIoT

S. Venkatasubramanian,[1] S. Raja,[2] V. Sumanth,[3] Jaiprakash Narain Dwivedi,[4] J. Sathiaparkavi,[5] Santanu Modak,[6] and Mandefro Legesse Kejela[7]

[1]*Department of Computer Science Engineering, Saranathan College of Engineering, Trichy 620012, Tamilnadu, India*
[2]*School of Mechanical Engineering, Vellore Institute of Technology, Vellore 632004, Tamilnadu, India*
[3]*Department of Computer Science Engineering, Presidency University, Bengalur 560064, Karnataka, India*
[4]*Department of Electronics & Communication Engineering, Lingayas Vidyapeeth, Faridbad 121002, India*
[5]*Department of Computer Science Engineering, Saranathan College of Engineering, Trichy 620012, Tamilnadu, India*
[6]*Department of Computer Science, Asutosh College, West Bengal 700026, India*
[7]*Department of Computer Science Engineering, Ambo University, Ambo, Ethiopia*

Correspondence should be addressed to Mandefro Legesse Kejela; mandefro.legesse@ambou.edu.et

Received 18 April 2022; Accepted 28 May 2022; Published 14 June 2022

Academic Editor: Punit Gupta

Detecting the breakdown of industrial IoT devices is a major challenge. Despite these challenges, real-time sensor data from the industrial internet of things (IIoT) present several advantages, such as the ability to monitor and respond to events in real time. Sensor statistics from the IIoT can be processed, fused with other data sources, and used for rapid decision-making. The study also discusses how to manage denoising, missing data imputation, and outlier discovery using preprocessing. After that, data fusion techniques like the direct fusion technique are used to combine the cleaned sensor data. Fault detection in the IIoT can be accomplished by using a variety of deep learning models such as PropensityNet, deep neural network (DNN), and convolution neural networks-long short term memory network (CNS-LSTM). According to various outcomes, the suggested model is tested with Case Western Reserve University (CWRU) data. The results suggest that the method is viable and has a good level of accuracy and efficiency.

1. Introduction

Connected gadgets are commonplace in the IoT, a computing paradigm that relies on ubiquitous Internet connectivity. These smart things can sense their surroundings, transmit, and analyze the data they collect from the environment and then return relevant details to their surroundings in a form that can be understood by humans. M2M technologies with applications in the automation industry make up a subset of the IIoT, which is a subset of the IoT. Improved production efficiency and long-term viability are two key benefits of the IIoT's [1, 2] introduction to the industry. For Industry 4.0, which is enabled by the integration of cloud technologies and cyber systems, a wide range of sensors are being installed around the industrial operational situation and tackled. Proactive maintenance and a reduction in unplanned downtime can be achieved by the use of [3–7] data analysis technologies.

If some measures are missing owing to network or hardware subjects in the IIoT, then we must have a working mechanism in place. The problem of value imputation becomes crucial when sensor data contain many missing values. High-frequency data collection results in large gaps between data points, and all measurements taken during that period are lost if the network goes down. When data are missing [8], it could be because of a sensor failure or a network failure, or because hackers have removed data with malicious intent while it is being collected, processed, stored, or sent. Filling in the missing values is a related research challenge that must be addressed to ensure that the imputed values are as close as feasible to the genuine values to analyze the data. The data collected are so diverse that to deal with missing data in IoT systems, the methodologies created must be able to provide a high level of confidence for various applications and endure the expanding deployments in the IoT (and IIoT) space. Additionally, real-time IoT application requirements require light-weight solutions [9].

Only data imputation, anomaly detection, and fault classification have been documented in the literature so far. Because each strategy may be maximized individually in this study, we can increase the monitoring system's overall performance by integrating all three techniques. There are three primary objectives for sensor networks in the IIoT, one being extracting relevant information for decision making [10, 11].

In the raw sensor data provided by IIoT sensors, there are a lot of unnecessary and uncleaned data. Consequently, to get any meaningful information from the cleaned IoT sensor data, the raw sensor data must be cleaned [12, 13]. A constrained IoT sensor network can also lead to high computational expenses and overuse of resources because of the vast amount of unwanted and worthless data [14–16].

2. Related Works

For chiller malfunction detection systems, Srinivasan et al. [15] showed the rank of understandable AI (XAI). One-dimensional convolutional neural networks (CNNs) were created by Li et al. [16] for defect identification in HVAC systems. Other M&E service systems have data-driven FDD approaches proposed in addition to the interpretability research for HVAC systems. Defect detection in sewerage systems was pioneered by Kumar et al. [17]. For picture object detection, the deep learning assembly trusts the CNN. In comparison to other machine learning (ML) techniques, the image processing skill employing the CNN is more understandable by experts. Gonzalez-Jimenez et al. [18] evaluated the existing fault diagnosis methods for electrical energies and re-examined the general process utilizing ML practices for electric drive fault detection. The lack of specific events in each electric drive is a major shortcoming of the data-driven FDD technique [18].

A microgrid's energy management system was studied by Marquez et al. [19], who used a fault detection and reconfiguration process. A reconfiguration block received fault information through acquiring residuals. Microgrid fault detection is all discussed in Morato et al. [20]. For early wind turbine breakdown detection, Ruiming et al. [21] proposed combining SCADA data and a dynamic network marker. Radial basis functions with two input parameters were used by Hussain et al. [22] to detect faults in solar systems [22]. Most solar energy applications are concerned with fault discovery in solar systems rather than fault diagnosis in solar power facilities. To better anticipate solar hot water system performance under various weather circumstances [23, 24], multiple deep learning models have been constructed to look for deviations between predictions and observations.

A significant advantage over prior approaches is the ability to isolate problems with the collectors' optical efficiency, flow rate, and thermal losses. In contrast to optical efficiency problems, which are caused by dirty or externally imposed collectors, deteriorating, breaking down, corroding, or otherwise degrading, problems with flow rate are caused by a loop that is out of balance, relative to the rest of the plant [25]. Assuming, as is the case in most real plants, that the only flow meter for the whole system is situated at the pump, and that thermal losses are caused by dirt, wear, insulation failures, and pipe breakage. However, the temperature reduction could be caused by an incorrect reading of the loop flow rate or by a broken pipe. Since the treatment for each case is different, it is important to know where the problem is to fix it quickly. Since a flow meter replacement is costly, it should only be done when there is confidence that this is the faulty component.

3. Proposed System

Consistently or in response to an external incident, IIoT sensors generate data. The other phase involves the collection, aggregation, analysis, and visualization of data generated by sensor nodes. This information is subsequently translated into a form that can be communicated as a response to an external stimulus. Data from IIoT sensors have the following notable properties:

Technical constraints: the sensor's small size imposes limitations on the sensor's computer power, storage capacity, and memory. Consequently, sensor data may be lost or incorrect information may be obtained if these devices are attacked or fail to operate.

Real-time processing: in future, the sensor network will be able to execute increasingly complicated networking activities and perform real-time data transformations from raw sensor data.

Scalability: the sensor network in the real world is made up of a variety of sensors and actuators. As the number of sensors and actuators continues to expand, the need for scalable sensor networks that can handle the increasing volume of data expands too.

Data representation: sensor data are often stored in the form of a tiny tuple containing structured information. Sensor data can be represented in a variety of ways, including Boolean, binary, featured, continuous, and numeric.

Heterogeneity: there is a wide range of data from IIoT sensors from rigorously formatted datasets to real-time information systems.

3.1. Denoising. IIoT networks create a lot of sensor data, which need to be analyzed and used for real-time decision-making. Sensor data have a wide range of features, including high velocities, large volumes, and a wide range of dynamic values and type values. They pollute and complicate decision-making in real time, as the sensor data are being collected and analyzed. Unwanted changes and modifications to the signal's original vectors are caused by noise, an uncorrelated signal component. To process and utilize the unusable data, resources are wasted because of the noise characteristic. It is possible to accurately characterize the signal using wavelet transform methods and to solve the problem of signal estimation using these methods. By reducing the signal's noise, the wavelet transformation keeps the original signal coefficients intact. This is accomplished through the use of a thresholding approach that is optimized for low-coefficient noise signals. To analyze and synthesize continuous-time signal energy, wavelet transformation is widely used.

To express the signal energy, we can use $e(t), t - R$.

$$|e|^2 = \int_{-\infty}^{\infty} |e(t)|^2 \mathrm{d}t < \infty. \tag{1}$$

In (1), it must be within the squared search space L2 for the signal energy $e(t)$ to satisfy the requirement (R). Analysis of discrete-time signal energy can also be done using the wavelet transformation.

3.2. Missing Data Imputation. When dealing with missing data, imputation is a necessary preprocessing step [26]. Various sectors and fields such as smart cities, healthcare, GPS, and smart transportation rely heavily on data generated by the internet of things (IoT). Algorithms for analyzing IIoT data typically presume that the data are completed before beginning their analysis. IoT data that are partial or missing can give unreliable results because of the data analytics conducted on that data. For the IIoT, an estimation of the missing value is required. As a first step, three things must be done. Identifying the cause of missing data is a critical first step. This is a result of poor network connectivity and defective sensor systems as well as environmental conditions and synchronization challenges. Data that are entirely missing at random (MCAR) are the most common sort of missing data, which is also the most difficult to find (NMAR). The next step is to look at the patterns of data that are missing. A random missing pattern is RMP and a monotonous missing pattern is AMP. As the last step, they create an IIoT for the missing datasets.

3.3. Data Outlier Detection. Sensor nodes in the IIoT sensor network are extensively dispersed and diverse. Note that, in a real-world physical context, such a design leads to significant sensor node failure and danger owing to a variety of outside influences. As a result, the IIoT sensor network's original data are vulnerable to manipulation, leading to data outliers [27]. Data outliers must be identified before data analysis or decision-making may take place.

Voting mechanism: an aberrant sensor node can be recognized in this manner through the comparison of its readings with those of neighboring sensors. According to Shahraki et al. [28], using a Poisson distribution to generate data in sensor network applications is the norm. Outliers for short-term, nonperiodic, and inconsequential variations in data patterns are generated in the IoT sensor network's data sets. Outliers in the IoT sensor network data with a Poisson distribution can be easily identified using the standard deviation and boxplot. The data generated by the failing sensor node and its nearby neighbors are also assessed using Euclidean distances in a distributed context. The data from this node is considered an outlier if the estimated variation exceeds a predetermined threshold. This approach, with its simplicity and convenience of use, relies heavily on the proximity of the sensor nodes to each other. The sparse network also has low precision.

3.4. Data Fusion. It is required to integrate or fuse data from several sensors to increase the accuracy of various applications. Sifting data from multiple sensors into an accurate, reliable, and trustworthy representation of the dynamic system's state is known as sensor fusion. This approximation is more accurate than using the sensors one at a time. Sensor fusion aims to lower the system's cost, complexity, and the number of parts while also improving the system's sensing precision and confidence. It is a multifaceted approach.

System states can include acceleration and distance from sensors or mathematical models. In addition to increasing the quality of data, the fusion of sensors can also boost the dependability, measure unmeasured states, and expand the coverage area.

Using this method, all sensors are connected and used to classify each other. A data-level fusion occurs at this point. When all of the sensors' data is combined, data features can be retrieved. Objects with sensors can be identified using these data properties. When numerous sensors' association identities are jointly proclaimed, this technique of direct fusion is also known as joint identity declaration. Equations (2)–(5) show the formal design of the direct fusion process.

$$O\colon S_i \longrightarrow S_j \forall \neq ji \in \{1, 2, 3, \ldots, n\}, \tag{2}$$

$$P\colon f_{f_e}\Big(f_{d_f}\left(f_A(O)\right)\Big), \tag{3}$$

$$ID_{\text{data}_{\text{extraction}}}(S_i) = g(p), \tag{4}$$

$$Q\colon JID_{\text{declaration}}(S_i). \tag{5}$$

The feature extraction result is subjected to the identity declaration function (g) in order to identify the specific sensor data (P). Finally, the function JID declaration (.) and the result Q are the outcomes of joint identification in the direct fusion strategy.

3.5. Fault Classification Using Proposed Model. An ensemble deep learning model for fault diagnosis uses these fused datasets as input.

3.5.1. Deep Neural Network (DNN). Artificial neural networks (ANNs) are a class of techniques that use stacks of layers to build a DNN model. Supervised learning can be used as unsupervised learning [29] as well. Weights in DNN models are stored in hidden layers. They are constantly being recalibrated during the training as they process new information. Finding more accurate patterns is why the weights are adjusted. The researcher does not need to indicate any patterns in advance for a DNN to learn. A subfield of machine learning known as "representation learning" (sometimes known as "feature learning") underpins deep learning techniques [30]. In contrast to machine learning algorithms, which need the researcher to manually select features before they can be employed, these approaches automatically select features.

There are four completely connected layers in the DNN architecture depicted in Table 1. According to [31], deep learning models are constructed by combining layers that are compatible and allow for effective data manipulations. Deep learning models can only take and produce outputs of a specific shape, which means that each layer of a model is limited to receiving and producing input tensors of one specific shape. According to [31], there is no need to be concerned about the connecting layers' suitability because they are created to match the geometry of the incoming layer. Tensor dimensions that are returned by a layer are known as its output shape. Table 1 demonstrates that the first hidden layer of DNN will return a tensor with dimensions of $n = 1$ (None, 64). The output form of this first hidden layer is (None, 64), and it has 64 neurons/units. The output shape from the first layer is automatically inferred as the input shape for the second layer. For a variable batch size, a dynamic dimension of a batch called a "mini-batch" (None) is utilized, allowing the user to specify any batch size for the deep neural network. Except for the most extreme circumstances, it is unnecessary to fix the first dimension of None at this time. During the fit or prediction phase, the batch size is determined.

To avoid models from overfitting, the dropout layers after the dense layers are utilized [32]. To stabilize the learning process and dramatically minimize training epochs, the design leverages batch normalization. The DNN learning process includes two important steps. The first stage is that the input layer delivers the raw data for the training data's forward propagation phase. As a second stage, the erroneous signal must be retransmitted. Neurons in the hidden layers process the data provided to the output layers to generate output data. Nonlinear functions are used to transfer the output data to the next layer. Activation functions refer to these nonlinear functions. The logistic function, the hyperbolic tangent, and the rectified linear unit ReLU are all examples of activation functions. An input signal from a DNN node is transformed into an output signal by these devices. This study employs the ReLU activation function, which significantly decreases training time and provides faster computation and convergence [33]. In deep learning, ReLU outperforms the sigmoid and tanh activation functions in terms of performance and generalization. DNN's final layer for multiclass models is a softmax layer, which keeps track of the probabilities associated with each class. The definition of the softmax layer used for K-class classification is as follows [34]:

Table 1: The architecture of DNN.

Type of layer	Number of parameters	Output shape
Dropout	0	(None, 32)
Dense	528	(None, 16)
Dense	768	(None, 64)
Dropout	0	(None, 64)
Dense	2080	(None, 32)
Batch normalization	64	(None, 16)
Dense	17	(None, 1)

$$f\left(x_i\right) = \frac{\exp\left(x_i\right)}{\sum_j \exp\left(x_j\right) \forall j \in \{1, \ldots, K\}.} \tag{6}$$

The *f*(xi) softmax function generates an output ranging from 0 to 1, with a probability total of 1. For binary classification, we will use a sigmoid function as the final layer to generate probabilities ranging from zero to one. Each new/test unit's propensity score is calculated using these probabilities.

3.5.2. *PropensityNet.*

According to [35], the PropensityNet (PN) is a deep neural network capable of predicting propensity scores. Table 2 shows that PN has five dense layers, which means they are all connected. To tackle a binary classification problem, PropensityNet uses Adadelta [36] as an optimizer and binary cross-entropy as an error measure. As the final layer, a sigmoid function is employed to provide probabilities ranging from 0 to 1. For each new/test unit, these probabilities serve as a measure of its propensity score. PropensityNet was built using Keras with a Tensorflow backend in *R* [37].

3.5.3. *CNN-LSTM.*

To learn long-term dependencies, long short-term memory networks (LSTMs) [38] have typically been used (subnets). Recurrent neural networks are an example of this type of network (RNNs). For nontemporal/sequential data, we employ a hybrid model that blends CNN and LSTM models. As a result, we check to see if the hybrid model can be tweaked to predict the probability of class membership (propensity scores). If we want to get probabilities (propensity scores) between zero and one, we will have to utilize a sigmoid function as the last layer. Li et al. [39] provide a thorough explanation of CNN-LSTM. Table 3 shows the hybrid CNN-LSTM model's design.

4. Results and Discussion

4.1. *Simulation Environment.*

Ubuntu 18.04, 32 GB of RAM, and an Intel i7-8700K CPU type powered our test PC. Ensemble deep learning models were developed mostly on the Python programming language.

4.2. *CWRU Dataset Introduction.*

Multiple accelerometers were placed around the bearing motor to collect vibration data in various operating conditions, resulting in the CWRU dataset from Case Western Reserve University Laboratory.

Table 2: The architecture of PN.

Type of layer	Number of parameters	Output shape
Dropout	0	(None, 32)
Dense	1056	(None, 32)
Dropout	0	(None, 32)
Dense	1056	(None, 32)
Dropout	0	(None, 32)
Dense	512	(None, 32)
Dropout	0	(None, 32)
Dense	1056	(None, 32)
Dense	33	(None, 1)

Table 3: The architecture of CNN-LSTM.

Type of layer	Number of parameters	Output shape
Conv1D	2080	(None, 15, 32)
Conv1D	4160	(None, 15, 64)
Maxpooling1D	0	(None, 15, 64)
Flatten	0	(None, 480)
Maxpooling1D	0	(None, 15, 32)
Dropout	0	(None, 15, 32)
LSTM	8320	(None, 15, 32)
Conv1D	128	(None, 15, 64)
Maxpooling1D	0	(None, 15, 64)
Dense	61568	(None, 128)
Dropout	0	(None, 128)
Dense	129	(None, 1)

In the CWRU dataset, the electric spark destroyed the bearing motor, replicating actual bearing failures. The inner raceway, the rolling element, and the outside raceway of the bearing driving end or fan end all had different fault locations. Various forms of fault could be represented by the bearing rolling element's four available diameters. Each of the four load types and four speeds of the bearing motor represented a different issue in the bearing motor. For the bearing motor depicted in Figure 1, see [39].

Experimental data were drawn from a subset of the CWRU dataset. Conditions for gathering normal state data included the motor load value of 1 horsepower and 1772 rpm speed value. It was determined that the frequency was 12000 samples per second and the diameter was 0.007 inches. The drive end had faults in three places: the rolling raceway, the outer raceway, and the inner raceway.

4.3. Evaluation Metrics.

There are many ways to evaluate the performance of a suggested approach for fault diagnostics. The Confusion Matrix, which is a two-dimensional matrix that provides information about the actual and predicted classes, serves as the basis for all evaluation criteria.

The right predictions are signified by the confusion matrix's diagonal members, while the incorrect guesses are represented by the confusion matrix's nondiagonal members. Confusion matrix attributes are depicted in Table 4.

Aside from that, recent research has employed a variety of evaluation metrics.

Precision: all samples projected as faults divided by the correct number of faults is represented by this ratio.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{7}$$

Recall: all samples accurately categorized as Faults are divided by the total number of samples that are in fact faults to get this ratio. Detection rate is another name for it.

$$\text{Recall} = \text{detection rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{8}$$

False alarm rate: also known as the false positive rate, it is the proportion of fault samples that are found to be normal.

$$\text{False alarm rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{9}$$

True negative rate: as a percentage of all samples that are categorized as normal, it is known as a normality index.

$$\text{True negative rate} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{10}$$

Accuracy: it is a measure of how many cases were correctly categorized out of all the ones that were found. When a dataset is balanced, the etection accuracy metric can be used as a helpful performance measure.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{11}$$

F-measure: to put it another way, it is the arithmetic mean of precision and recall combined. Accuracy may be evaluated by looking at both the system's precision and recall, which is what this technique is all about.

$$F - \text{measure} = 2\left(\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right). \tag{12}$$

4.4. Performance Evaluation of the Proposed Model.

Here, the performance of the proposed ensemble model is tested with two different training sets and testing sets. Initially, 50% of training data and 50% of testing data are considered for validation, which is shown in Table 5 and Figure 2 presents the comparative analysis of the proposed model with different techniques in terms of accuracy for different data ratios.

In the analysis of accuracy, CNN, DNN, and CNN-LSTM achieved 92%, LSTM and RNN achieved nearly 85%, PropensityNet achieved 93%, and the proposed model achieved 94.32%. The single classifiers achieved less performance, but when it is in ensemble state, they achieved better performance. The reason is that training data are accurate and the prediction of faults on each technique is accurate and it is finalized effectively. However, the accuracy of performance and recall is low even for the proposed ensemble model (i.e., 93.24% of recall) and this is because only 50% of data is trained and tested with the remaining 50% of data. In addition, some data are missed while training the model. But, in the analysis of precision, all techniques achieved nearly 88% to 90% and achieved nearly 92% to 96% of F-score. The next set of experiments is carried out by considering 75% of training data and 25% of testing data, which is shown in Table 6.

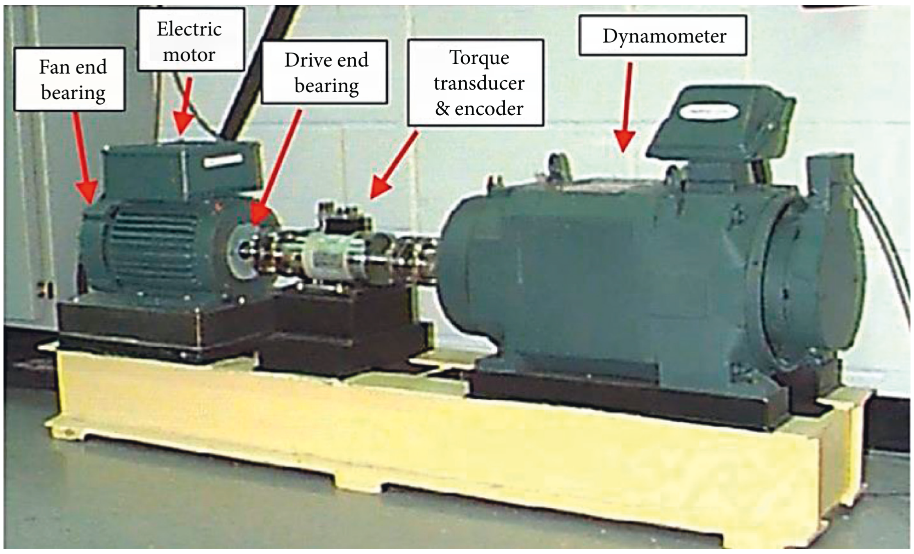

Figure 1: Bearing motor setup to generate CWRU dataset.

Table 4: Confusion matrix.

		Predicted class	
		Fault	Normal
Actual class	Normal	False positive	True negative
	Fault	True positive	False negative

Table 5: Comparative analysis of 50%–50% on the proposed model with various existing algorithms.

Algorithm	Accuracy	Precision	Recall	F-score
RNN	81.10	89.41	75.91	86.72
LSTM	87.70	89.41	85.21	91.82
CNN	92.50	89.82	90.98	95.27
DNN	92.50	87.63	91.38	95.18
CNN-LSTM	92.70	88.90	91.93	95.32
PropensityNet	93.27	87.91	92.47	95.63
Proposed ensemble	94.32	90.95	93.24	96.02

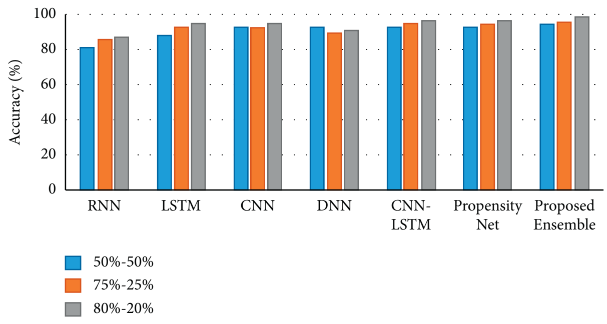

Figure 2: Graphical representation of the proposed model in terms of accuracy for different training and testing data ratios.

When the training data is increased, the performance of the models is increased; for instance, the proposed ensemble model achieved 95.62% of accuracy, 98.32% of precision, 94.62% of recall, and 94.53% of F-score. When compared with all techniques, RNN achieved low performance, i.e., 85.71% of accuracy, 84.32% of precision, 85.93% of recall, and 83.45% of F-score. The reason for poor performance is that RNN takes a long time to train the network and it is inefficient to handle the missing data. Moreover, the raw data are fused by direct fusion techniques, and then it is used for identifying the faults in machines. All proposed single classifiers such as DNN, CNN-LSTM, and PropensityNet achieved nearly 94% of accuracy, 96% of precision, 92% of recall, and an F-score. LSTM and CNN models achieved 92% of accuracy, 93% of precision, 92% of recall, and 91% of F-score. The training set is increased to 80% and testing data is set at 20%, which is shown in Table 7.

Table 6: Comparative analysis of 75%–25% on the proposed model with various existing algorithms.

Algorithm	Accuracy	Precision	Recall	F-score
RNN	85.71	84.32	85.93	83.45
LSTM	92.10	92.43	92.15	91.68
CNN	92.46	93.48	92.44	91.81
DNN	89.52	90.21	89.54	89.03
CNN-LSTM	94.53	96.61	92.52	92.24
PropensityNet	94.16	96.17	92.32	92.10
Proposed ensemble	95.62	98.32	94.62	94.53

Most of the techniques' recall and F-score are nearly the same; for instance, RNN achieved 81%, LSTM achieved 88%, CNN achieved 92%, DNN achieved 92%, CNN-LSTM achieved 91%, PropensityNet achieved 92%, and the proposed ensemble model achieved 93%. The accuracy of all techniques is increased, when compared with the first set of experiments. That is, the proposed model achieved 98.84% of accuracy, CNN-LSTM and PropensityNet achieved 96%, and LSTM and CNN achieved 94% of accuracy. The DNN has 93.45% of precision, CNN-LSTM achieved 97.85% of precision, PropensityNet achieved 98.23% of precision, and the ensemble model achieved 99.23% of precision.

Table 7: Comparative analysis of 80%–20% on the proposed model with various existing algorithms.

Algorithm	Accuracy	Precision	Recall	F-score
RNN	87.35	78.21	81.10	81.11
LSTM	94.82	89.33	87.70	88.64
CNN	94.62	88.82	92.50	89.02
DNN	90.76	93.45	92.50	87.39
CNN-LSTM	96.63	97.85	92.70	90.33
PropensityNet	96.12	98.23	93.27	91.02
Proposed ensemble	98.84	99.23	94.32	92.91

5. Conclusion

As artificial intelligence technology continues to advance, it is now possible to foresee mechanical failures based on the IIoT. Sensor data fusion knowledge relies on big data processing and analysis. Models and methods for sensing data fusion in defect detection and prediction were examined in this research. The direct fusion model is provided here in terms of fusion models. To train and retrieve the original data, the relevant ensemble methods based on deep learning can be immediately implemented. Data preprocessing is not usually necessary, but the learning curve was steep and the machine performance needs were high. Because of this, the preprocessing stage includes missing data, outlier detection, and data imputation. Results from the trials demonstrate that the suggested ensemble model achieved 94% accuracy on 50%–50% of data, 95.6% accuracy on 75% to 25% of data, and 98% accuracy on 80%–20% of data, where the single DL models achieved approximately 96% accuracy on 80%–20% data.

6. Limitation and Future Scope

The following are the obstacles and difficulties encountered in the context of fusing sensory data, based on the current development state of fusion models:

(1) Fusion models are not all the same: there is no one-size-fits-all model for mechanical defect diagnosis and prediction in the field. A large number of current fusion models are based on a certain type of device. Developing a common framework for identifying mechanical equipment failures in the future would be advantageous.

(2) Uncertainty in the original data: during the data gathering process, a lot of noise is present in the actual data obtained since environmental elements cannot be controlled. Data fusion and feature extraction are often incorrect if the unique data are used directly. It is therefore vital to select a suitable data preprocessing approach instead of techniques utilized in this study when raw data are given. A set of preprocessing methods for diverse sensors used in fault analysis and prediction for mechanical gear will be beneficial in the future development process.

(3) Long running time: finding appropriate hyper parameters requires a lot of running time when using fusion methods based on deep learning. Overfitting can also occur. Fusion techniques typically necessitate feature extraction by hand, which adds time to the computation time. Research into the feature and decision-level fused algorithms are the focus of the majority of fusion algorithms. There are very few data fusion algorithms. As a result, it will be necessary to continue working on data fusion algorithms in the future.

Data Availability

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy concerns.

Conflicts of Interest

The authors declare that they have no conflicts of interests.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, “Industrial Internet of things: challenges, opportunities, and directions,” *IEEE Internet of Things Journal*, vol. 14, 2018.

[2] Ericsson, “Cellular networks for massive IoT,” 2020, https://www.ericsson.com/assets/local/publications/whitepapers/wp%20iot.pdf.

[3] Y. Liu, T. Dillon, W. Yu, W. Rahayu, and F. Mostafa, “Missing value imputation for industrial iot sensor data with large gaps,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6855–6867, 2020.

[4] F. Civerchia, S. Bocchino, C. Salvadori, E. Rossi, L. Maggiani, and M. Petracca, “Industrial internet of things monitoring solution for advanced predictive maintenance applications,” *Journal of Industrial Information Integration*, vol. 7, pp. 4–12, 2017.

[5] J. Wan, S. Tang, D. Li et al., “A manufacturing big data solution for active preventive maintenance,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2039–2047, 2017.

[6] B. Cheng, J. Zhang, G. P. Hancke, S. Karnouskos, and A. W. Colombo, “Industrial cyberphysical systems: realizing cloud-based big data infrastructures,” *IEEE Industrial Electronics Magazine*, vol. 12, no. 1, pp. 25–35, 2018.

[7] W. Yu, T. Dillon, F. Mostafa, W. Rahayu, and Y. Liu, “A global manufacturing big data ecosystem for fault detection in predictive maintenance,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 183–192, 2020.

[8] P. Nardelli, C. Papadias, C. Kalalas et al., “Framework for the identification of rare events via machine learning and iot networks,” in *Proceedings of the 2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, pp. 656–660, IEEE, Oulu, Finland, August, 2019.

[9] A. Gonzalez-Vidal, P. Rathore, A. Rao, J. Mendoza-Bernal, M. Palaniswami, and A. Skarmeta-Gomez, “Missing data imputation ´ with bayesian maximum entropy for internet of things applications,” *IEEE Internet of Things Journal*, vol. 8, 2020.

[10] X. Deng, P. Jiang, X. Peng, and C. Mi, “An intelligent outlier detection method with one class support tucker machine and genetic algorithm toward big sensor data in internet of things,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4672–4683, 2019.

[11] S. Sanyal and P. Zhang, “Improving quality of data: IoT data aggregation using device to device communications,” *IEEE Access*, vol. 6, Article ID 67830, 2018.

[12] D. B. Rubin, “Inference and missing data,” *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.

[13] Y. Dong and C.-Y. J. Peng, “Principled missing data methods for researchers,” *SpringerPlus*, vol. 2, no. 1, p. 222, 2013.

[14] C. Cunqing Hua and T.-S. P. Yum, “Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 892–903, 2008.

[15] S. Srinivasan, P. Arjunan, B. Jin, A. L. Sangiovanni-Vincentelli, Z. Sultan, and K. Poolla, “Explainable AI for chiller fault-detection systems: gaining human trust,” *Computer*, vol. 54, no. 10, pp. 60–68, 2021.

[16] G. Li, Q. Yao, and C. Fan, “An explainable one-dimensional convolutional neural networks based fault diagnosis method for building heating, ventilation and air conditioning systems,” *Building and Environment*, vol. 203, Article ID 108057, 2021.

[17] S. S. Kumar, D. Abraham, and M. Rosenthal, “Leveraging visualization techniques to develop improved deep neural network architecture for sewer defect identification,” in *Construction Research Congress 2020: Infrastructure Systems and Sustainability*, pp. 827–835, American Society of Civil Engineers, Reston, VA, USA, 2020.

[18] D. Gonzalez-Jimenez, J. del-Olmo, J. Poza, F. Garramiola, and P. Madina, “Data-driven fault diagnosis for electric drives: a review,” *Sensors*, vol. 21, no. 12, p. 4024, 2021.

[19] J. J. Marquez, A. Zafra-Cabeza, C. Bordons, and M. A. Ridao, “A fault detection and reconfiguration approach for mpc-based energy management in an experimental microgrid,” *Control Engineering Practice*, vol. 107, Article ID 104695, 2021.

[20] M. M. Morato, P. R. C. Mendes, and J. E. Normey-Rico, “Dealing with energy- generation faults to improve the resilience of microgrids: a survey,” in *Proceedings of the 2019 IEEE PES Innovative Smart Grid Technologies Conference - Latin America*, p. 1e6, September, 2019.

[21] F. Ruiming, W. Minling, G. xinhua, S. Rongyan, and S. Pengfei, “Identifying early defects of wind turbine based on scada data and dynamical network marker,” *Renewable Energy*, vol. 154, pp. 625–635, 2020.

[22] M. Hussain, M. Dhimish, S. Titarenko, and P. Mather, “Artificial neural network based photovoltaic fault detection algorithm integrating two bi-directional input parameters,” *Renewable Energy*, vol. 155, pp. 1272–1292, 2020.

[23] C. Correa-Jullian, J. M. Cardemil, E. L. Droguett, and M. Behzad, “Assessment of deep learning algorithms for fault diagnosis in solar thermal systems,” in *ISES Solar World Congress*, 2019.

[24] C. Correa-Jullian, J. M. Cardemil, E. López Droguett, and M. Behzad, “Assessment of deep learning techniques for prognosis of solar thermal systems,” *Renewable Energy*, vol. 145, pp. 2178–2191, 2020.

[25] A. J. Gallego, M. Macías, F. d. de Castilla, and E. F. Camacho, “Mathematical modeling of the Mojave solar plants,” *Energies*, vol. 12, no. 21, p. 4197, 2019.

[26] X. Yan, W. Xiong, L. Hu, F. Wang, and K. Zhao, “Missing value imputation based on Gaussian mixture model for the internet of things,” *Mathematical Problems in Engineering*, vol. 2015, pp. 1–8, 2015.

[27] A. A. Al-khatib, B. Mohammed, and K. Abdelmajid, “A survey on outlier detection in internet of things big data,” in *Big Data-Enabled Internet of Things*, pp. 265–272, IET, London, UK, 2020.

[28] A. Shahraki, Ø. Haugen, and Ø. Haugen, “An outlier detection method to improve gathered datasets for network behavior analysis in IoT,” *Journal of Communications*, vol. 14, pp. 455–462, 2019.

[29] A. Sivaram, L. Das, and V. Venkatasubramanian, “Hidden representations in deep neural networks: Part 1. Classification problems,” *Computers & Chemical Engineering*, vol. 134, Article ID 106669, 2020.

[30] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: a review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[31] F. Chollet, *Deep Learning with Python*, Simon & Schuster, New York, NY, USA, 2021.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “‘Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, 2014.

[33] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning,” 2018, https://arxiv.org/abs/1811.03378.

[34] V. Ramachandra, “Deep Learning for Causal Inference,” 2018, https://arxiv.org./abs/1803.00149.

[35] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” 2012, https://arxiv.org/abs/1212.5701.

[36] F. Chollet and J. Allaire, “R interface to Keras,” 2017, https://github.com/rstudio/keras.

[37] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] A. Whata and C. Chimedza, “Deep learning for SARS COV-2 genome sequences,” *IEEE Access*, vol. 9, Article ID 59597, 2021.

[39] S. Li, G. Liu, X. Tang, J. Lu, and J. Hu, “An ensemble deep convolutional neural network model with improved D-S evidence fusion for bearing fault diagnosis,” *Sensors*, vol. 17, no. 8, p. 1729, 2017.