Computer Communications 187 (2022) 164–171

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A sustainable deep learning framework for fault detection in 6G Industry 4.0 heterogeneous data environments

Tinhinane Mezair [a], Youcef Djenouri [b], Asma Belhadi [c], Gautam Srivastava [d,e], Jerry Chun-Wei Lin [f,*]

[a] *Ecole Nationale Polytechnique, Algiers, Algeria*
[b] *SINTEF Digital, Oslo, Norway*
[c] *Kristiania University College, Oslo, Norway*
[d] *Brandon University, Brandon, Canada*
[e] *China Medical University, Taichung, Taiwan*
[f] *Western Norway University of Applied Sciences, Bergen, Norway*

A R T I C L E I N F O

Keywords:
Industry 4.0
Deep Learning
Fault Detection
Machine to machine communication
Sustainable 6G

A B S T R A C T

The integration of 5G and Beyond 5G (B5G)/6G in Machine-to-Machine (M2M) communications, is making Industry 4.0 smarter. However, the goal of having a sustainable self-monitored industry has not been reached yet. State-of-the-art deep learning-based Fault Detection algorithms cannot handle heterogeneous data, meaning that more than one fault detection computational device has to be used for each data format, in addition to the inability to take advantage of the combination of all the information available in different formats to derive more accurate conclusions. Moreover, these algorithms rely on inefficient hyper-parameters tuning strategies. In this paper, we propose an Advanced Deep Learning framework for Fault Diagnosis in Industry 4.0 (ADL-FDI4), which combines Long Short Term Memory (LSTM), Convolutional Neural Networks (CNN) and graph CNN (GNN), to handle heterogeneous data. Furthermore, our novel framework uses a Branch-and-Bound procedure to guide the learning process. Our experimental results show that ADL-FDI4 outperforms the state-of-the-art solutions in terms of detection rate and running time, and for that, it consumes less energy. In addition to handling heterogeneous data, which implies that one computational device is sufficient to handle all data formats.

1. Introduction

The developments in 5G and Beyond 5G (B5G)/6G technologies and the start of their deployment lead to huge advances in the Internet of Things (IoT) and Machine-to-Machine (M2M) communications. This has benefited industrial automation, increased productivity, reduced production costs, and created new industrial ecosystems (Industry 4.0) [1–7].

The Fourth Industrial Revolution (Industry 4.0) is a concept of major interest to the business, industrial, and academic research communities since 2014 [8–10]. Researchers are developing new technologies to make this new industry smart, autonomous, and sustainable. Thanks to the enabled ubiquitous sensing and actuation in distributed IoT platforms with the integration of (B5G)/6G, fault detection is currently a hot research topic that concerns itself with the identification and localization of faults in systems and processes.

However, the state-of-the-art deep learning-based fault detection algorithms cannot handle heterogeneous data, while the data collected from the various sensors and smart devices can be of different formats, it can for instance be images, videos, time series, or even graphs. This might lead to the necessity of using a lot of fault detection computational devices to handle all the received data formats. In addition to this, it is very common to have different types of sensor data and information available about an industrial event; and taking advantage of the combination of these different data formats may significantly help derive a more accurate classification of the observation/event. In this paper, we also show that the state-of-the-art algorithms are slow and therefore require more computational resources than necessary. Also, these algorithms rely on exhaustive search strategies for hyper-parameter tuning in the training phase, which is not the most efficient

* Corresponding author.
E-mail addresses: tinhinane.mezair@g.enp.edu.dz (T. Mezair), youcef.djenouri@sintef.no (Y. Djenouri), asma.belhadi@kristiania.no (A. Belhadi), srivastavag@brandonu.ca (G. Srivastava), jerrylin@ieee.org (J.C.-W. Lin).

https://doi.org/10.1016/j.comcom.2022.02.010
Received 20 August 2021; Received in revised form 26 November 2021; Accepted 14 February 2022
Available online 22 February 2022

strategy. In this work, we propose an Advanced Deep Learning framework for Fault Diagnosis in Industry 4.0 (ADL-FDI4) to address these issues.

1.1. Motivations

Fault detection is a very important area of research in the 6G industry, especially when considering faults resulting from cyber-attacks; which we imagine are of primordial concern in such connected environments, and the risks they can represent. But the process of detecting faults can be costly and far from being sustainable, in addition to not being very reliable. In this modern industry, it is usually the case that data is very much available, the only problem is that it is usually available in different formats and not fully exploited. Using different data formats, and thus combining as much as possible of the information we can obtain about the same industry event, can greatly improve the classification accuracy of that event into a particular faultt category, or not into a fault. And improving the learning process by better choosing the hyper-parameters can significantly decrease the running time, and thereby make the model more sustainable.

1.2. Contributions

The main contributions of this research are listed as follows:

1. We propose a novel framework, named ADL-FDI4 (**A**dvanced **D**eep **L**earning framework for **F**ault **D**iagnosis in **I**ndustry **4**.0). This framework takes as input different types of data formats about the same industrial event: images, videos, time series, and graphs, and combines all the information available in these different formats to output the most accurate set of fault diagnoses about this event that is possible to get with the inputs. The model relies on the combination of LSTMs, CNNs and graph CNNs, to handle the heterogeneous data.
2. We propose a branch-and-bound optimization strategy for hyperparameters tuning of the different deep learning architectures involved in the model. The strategy considers the hyperparameters space and intelligently explores the enumeration tree using a heuristic instead of the exhaustive search-based methods.
3. For performance evaluation purposes, our model is compared in terms of accuracy and running time to the state-of-the-art fault detection algorithms in Industry 4.0 (B5G)/6G environments, using four benchmarks.

1.3. Outline

The rest of this paper is organized as follows. Section 2 gives an in-depth related work-study in fault diagnosis for Industry 4.0 in a (B5G)/6G environment. Section 3 presents a detailed explanation of the ADL-FDI4 framework. A performance evaluation of the ADL-FDI4 framework is provided in Section 4, which is followed by the conclusion in Section 5.

2. Literature review

2.1. Related studies

Jeon et al. [11] developed a smart machine software for Industry 4.0 in a systematic way instead of ad-hoc technologies. It allows to check all stakeholders' constraints, and implement them into automatic operations. It also incorporates the digital twin technology [12] which considerably helps the manufacturing process in visualizing the detected anomalies. Kiangala et al. [13] transformed the multivariate time series data collected from Industry 4.0 sensors to the set of images for predictive maintenance using the Gramian Angular Field [14]. The obtained images are trained using the convolution neural network with an adapted rectifier linear unit activation function. Dalzochio et al. [15] established a systematic analysis of 38 relevant research works in the applications of artificial intelligence techniques for predictive maintenance. This study reveals that the heterogeneity and computational resources are open challenges for predictive maintenance in Industry 4.0. Silvestri et al. [16] studied the interaction among the maintenance tasks and the different predictive maintenance and fault diagnosis strategies in Industry 4.0 context. They argued that the existing solutions for Industry 4.0 missed the incorporation of artificial intelligence and visualization techniques for better predictive maintenance operations. Kufner et al. [17] dealt with a large amount of data in production, and suggested a vertical-based solution while merging sensors data in a distributed cloud environment. Kaupp et al. [18] identified various kinds of contextual faults in a log of data collected from sensors in a smart factory system. The first kind is a compressor failure where lack of air pressure will harm the manufacturing process without stopping its functionalities. The second kind of contextual fault is the high delay of production time by doing a manual inspection. The third kind of contextual fault is performed during the pressing stage where the behavior of the manufacturing process is slightly changed once the assembling operator is executed. Natesha et al. [19] employed fog computing in the industrial internet of things environment for fault diagnosis. The industrial controller is used to process abnormal machine sounds. The malfunctioning machines are observed by the supervised machine learning architectures by training the linear prediction coefficients. Rahman et al. [20] developed a privacy-preserving solution to handle microservices in educational settings. The approach used blockchain strategy which aims to ensure integrity and confidentiality over different entities in the education system. The fault diagnosis process is also used to identify abnormal behaviors from microservices data. Long et al. [21] explored the visual features for motor fault diagnosis. The correlation between the anomalies and the visual features is studied using a matching process. This method is a fly-based solution where only the training data with small sizes is required. Liu et al. [22] suggested the use of distributed artificial intelligence to assist Industrial Internet of Things (IIoT) systems in predicting faults and anomalies. The feature selection is performed to reduce the searching space and execute the predictive maintenance framework in real-time. Yu et al. [23] proposed a multi-objective algorithm for handling imbalanced data of fault diagnosis scenarios. An adaptive loss function based on various misclassification metrics was used to learn the minority population. Hazra et al. [24] investigated the use of reinforcement learning for decision making in the Industrial Internet of Things (IIoT). The developed model can easily learn the different rules for controlling industrial networks, and then be able to deal with multi-user requests. Hazra et al. [25] improved the previous solution by developing the provisioning-based approach to process multiple fog devices. A task partitioning policy was also suggested. Adhikari et al. [26] reviewed several types of intrusion detection on the internet of vehicles systems. It also suggested some solutions to mitigate such attacks. Adhikari et al. [27] improved the previous solution by developing a reinforcement learning-based process. The support vector machine was also used to analyze the CyberTwin data.

2.2. Discussion

From this literature review, we can see that the existing technologies for fault diagnosis have several drawbacks. The first one is that they are not able to handle heterogeneous data with different data formats. While different types of data can be collected from the sensors in the Industry 4.0 environment, such as images, videos, and also graphs and time series. The second issue is that they rely on exhaustive search strategies for hyper-parameters optimization, where many parameters should be fixed and tuned in the training phase. In the next section, we present a new Advanced Deep Learning framework for Fault Diagnosis in Industry 4.0 (ADL-FDI4) to address the drawbacks mentioned above.

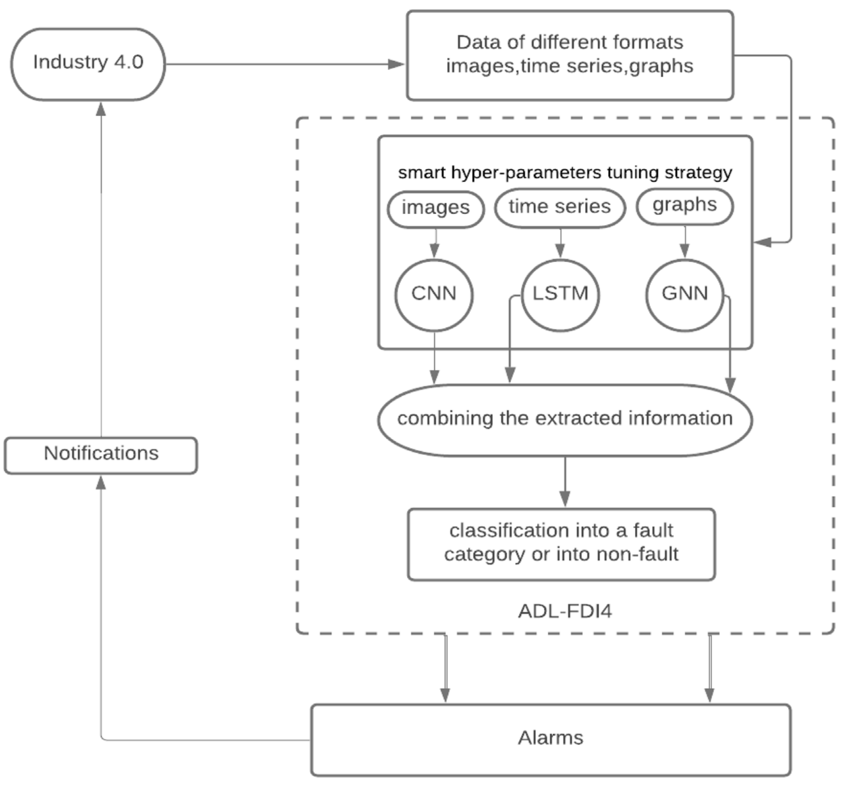
Industry 4.0
Data of different formats images,time series,graphs
smart hyper-parameters tuning strategy
images
time series
graphs
CNN
LSTM
GNN
combining the extracted information
classification into a fault category or into non-fault
ADL-FDI4
Notifications
Alarms

Fig. 1. ADL-FDI4 framework.

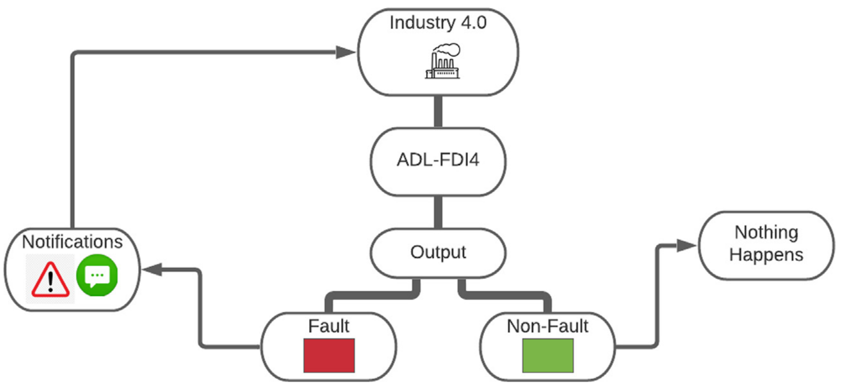

Fig. 2. ADL-FDI4 simplified visualization.

3. ADL-FDI4: advanced deep learning for fault diagnosis in Industry 4.0

3.1. Principle

Let us begin by introducing the main components of ADL-FDI4. Fig. 1 illustrates the designed framework which is based on different smart technologies, namely, Branch-and-Bound and different Deep Learning architectures, which are: Long Short Term Memory (LSTM), Convolutional Neural Networks (CNN), and graph CNN, to deal with Industry 4.0 data. The data is first extracted from different sensors. The deep learning is then executed to detect the faults in the system and trigger alarms which will notify the Industry 4.0 monitoring system of the detected faults. In this context, different deep learning architectures are combined to handle the heterogeneous data: long short-term memory for time series, a convolution neural network for images, and a graph convolution neural network for graph data. A Branch-and-Bound optimization strategy is proposed for the hyper-parameters tuning of the different deep learning models. The strategy considers the hyper-parameters space and intelligently explores the enumeration tree using a heuristic instead of the exhaustive search-based methods. Fig. 2 shows a simplified visualization of the model developed in this paper. In the remainder of this section, we describe the detailed ADL-FDI4 components.

3.2. ADL-FDI4 components

3.2.1. Long Short Term Memory

LSTM is the most known and most used recurrent neural network (RNN) architecture. It is very effective in learning from sequential and time-series data, therefore it is very used in Industry 4.0 applications. LSTM was specially designed to deal with the exploding/vanishing gradient problems that we may encounter when training traditional RNN. The LSTM architecture proposed in this research work is composed of four units:

1. **The prediction unit:** Is a neural network that receives an input vector and predicts an output vector. The prediction gets better the more the whole architecture is trained.
2. **An attention mechanism:** Is another neural network that is trained separately to learn to ignore some predictions (that will come from the first unit) that are not relevant at the moment, so they do not cloud the predictions in memory when going forward.
3. **A memory unit:** Another neural network that learns what to keep in memory and what to forget. Memorized information can then be combined with the coming predictions from the two first units to make a better prediction based on previously seen data.
4. **A selection unit:** Another neural network trained to filter and select only the final prediction to send as output and keep the memories in.

There are two types of gates that make all of this work together:

- The element by element addition gate.
- The element by element multiplication gate.

3.2.2. Convolution neural network

CNN is used in computer vision to handle images or videos. The main idea behind a convolution neural network is to filter each input image to extract certain features (the features that help the most in distinguishing the object we are looking for) before training the fully connected layers. To filter, we use what is called filter matrices (or filters). A filter is a set of multipliers; for example, if we have a filter matrix F of size 3×3, and the matrix M of the pixels of an image, we compute the new value of a pixel x in M by building a matrix A of its neighbors (x is in the center of this matrix A of size 3×3), and then we compute the element-wise product of F and A, the new value of x is the sum of all the products we obtained; we do this for all the elements of M to obtain the filtered image matrix. These filters are learned in the convolutional layer of a convolution neural network. In this layer a pre-selected number of randomly initialized filters will pass over each input image, the results are sent to the neural network and this is how the best filters are learned. This process is called feature extraction. The architecture of a convolution neural network can also contain pooling layers. Pooling is a way of compressing an image, by grouping some pixels in the image in sets and replacing each set with a subset of pixels. For example, max-pooling 2×2 groups the image into sets of 2×2 pixels and replaces each set by the largest pixel it contains, the image will then be reduced to a quarter of its original size.

3.2.3. Graph convolution neural network

Graph Convolution Neural Network (GNN) is a deep learning architecture that operates on graph-structured data. We want to take advantage of the convolutional layer of a CNN to operate on arbitrary graphs (graphs of any structure, cyclic or not, and with any number of nodes and edges) instead of images. We can see images as "grid graphs" (each node is a pixel, and the pixels' matrix of the image is the adjacency matrix of the grid graph that this image is). To apply the same idea of filtering an image on graphs; instead of having a pixel for which we use the information contained in its neighboring pixels to update its value, we have a node for which we use its neighboring nodes to update its features.

In a GNN, we can either classify each node independently, or classify the graph as a whole, or we can classify the edges or investigate the existence of a link between two nodes. To build a GNN, we start by constructing the adjacency matrix A of the graph: For example, in a non oriented graph we can take $A_{ij} = 1$ (with A_{ij} being an element of the adjacency matrix A, also called the neighboring matrix) if there is a link between node i and node j, and $A_{ij} = 0$ if i and j are not linked. We also construct the node matrix H, which contains the message or information stored in each node, and then build the matrix $H' = \sigma\left(\widehat{D}^{-1}\hat{A}HW\right)$ where W is a learnable node-wise shared linear transformation (which is a linear layer in a deep learning framework), σ is a nonlinear function such as ReLU, $\hat{A} = A + I$ (this matrix $\hat{A}$ is included to not discard the central node, it enforces that a node is always connected to itself), $\widehat{D}$ is the degree matrix, this matrix gives the degree of each node, $\widehat{D}^{-1}$ is integrated into the equation to normalize the adjacency matrix and force the features not to explode when summing. This is called the mean-pooling update rule. If we use the symmetric normalization:

$$H' = \sigma\left(\widehat{D}^{-1/2}\hat{A}\widehat{D}^{-1/2}HW\right) \tag{1}$$

We obtain the graph convolutional network (GCN) update rule. This is currently the most popular graph convolutional layer. Or in a more generalized representation, nodes can send arbitrary messages along the edges $\overrightarrow{e_{ij}}$, a node then aggregates all the messages it receives using a permutation-invariant function (such as a sum).

Let $\overrightarrow{m_{ij}}$ be the message sent from node i to node j, calculated using a message function f_e:

$$\overrightarrow{m_{ij}} = f_e\left(\overrightarrow{h_i}, \overrightarrow{h_j}, \overrightarrow{e_{ij}}\right), \tag{2}$$

then all messages entering a node are aggregated using a readout function as:

$$f_v : \overrightarrow{h_i'} = f_v\left(\overrightarrow{h_v}, \sum_{j\in N_i}\overrightarrow{m_{ji}}\right), \tag{3}$$

where N_i is the set of the neighbors of node i.

This gives the message-passing neural network (MPNN), that is in practice only applicable to small graphs. f_e and f_v are usually small multilayer perceptrons. A more general form is:

$$\overrightarrow{h_i'} = \sigma\left(\sum_{j\in N_i}\alpha_{ij}W\overrightarrow{h_j}\right), \tag{4}$$

where α_{ij} is a coefficient that is either defined explicitly which causes some shortcomings, or

$$\alpha_{ij} = \frac{\exp\left(a_{ij}\right)}{\sum_{k\in N_i}\exp\left(a_{ik}\right)}, \tag{5}$$

where

$$a_{ij} = a\left(\overrightarrow{h_v}, \overrightarrow{h_j}, \overrightarrow{e_{lj}}\right), \tag{6}$$

in which a is a learnable, shared, self-attention mechanism. This is called the graph attention network update rule.

In short, a given graph is encoded in three matrices: A (the adjacency matrix), H (the node matrix), D (the degree matrix), and a scalar W. With these parameters, a matrix H' is calculated following the chosen update rule equation. Feeding this graph as an input to the GNN means providing a deep neural network with this matrix H' as an input, for it to output a vector of the classification of the input, just as in any other deep neural network that gets a matrix as an input. The specification in a GNN is in the manner of encoding the graph into a matrix H'.

3.3. Branch-and-Bound for hyperparameters optimization

Hyper-parameters optimization is a primordial task when developing deep learning models. In this research work, we used a Branch-and-Bound (BB) strategy to intelligently optimize the different hyper-parameters of the different deep learning architectures used in our model. BB is a well-known algorithm for discrete optimization problems, combinatorial optimization problems and also for mathematical optimization. Where we need to keep track of what the lowest bound yet found is, to compare it with the possible solutions and only keep a possible solution and consider it the new lowest bound if it is inferior to the lowest bound yet found. From here we can deduce that we can only solve minimization problems, but any maximization problem can be formulated as a minimization problem by multiplying the objective function by −1. The only problems for which we can guarantee to find the global optimum are the convex problems. Branching consists of forming a rooted tree of the possible solutions to the problem. Then we can do an exhaustive search (explore all the branches of the tree), or get rid of some branches that we know cannot be solutions and not explore them (this is called pruning). When we have a convex problem, we can prune if we have one of these scenarios (if the problem is not convex, these conditions may lead to pruning too early and maybe miss a local optimum or miss the global optimum) :

1. Infeasibility of the value of a variable (a constraint). So we get rid of all the branches that are linked to that value. Because by going down we are only adding more constraints so if one is already infeasible no need to go further.
2. We have reached an objective that is worse than the best solution.
3. A solution is found. Going down cannot make us find a better solution (because we are just adding more constraints). Now we just compare this solution to the best one yet found.

There remains concerns on which variable to branch on. We give the example of a binary problem (a binary problem here is an optimization problem where the vector X of the variables x_i for $i = \{1, 2, 3, \dots, n\}$ is such that $X \in \{0, 1\}^n$ where n is the number of variables in the problem) relaxed to $0 \leq x_i \leq 1$ for $i = \{1, 2, 3, \dots, n\}$ that first gives this solution : $X = [0.1, 0.6, 1, 0, 1]$, here we better branch on the variable for which the value is the closest to 0.5 (in this example it is x_2). In non binary problems it is more difficult to decide.

There are two main exploration strategies: *depth first* (walk a branch to the bottom until we cannot go any further and then work our way up). Or *breadth first* (at each depth explore different branches and then go deeper and do the same again until we reach the bottom). Usually, *depth first* is a better strategy, because forcing more and more constraints usually forces a solution faster, and once we have a solution we can compare it to others and this helps us prune faster.

4. Performance evaluation

In this section, we provide an intensive numerical analysis of the proposed solution, comparing it with different baseline methods, using different data sources. In addition, a well-known evaluation measure is used to compute the accuracy performance. In the following section, we present the experimental settings used for the evaluation of the model.

4.1. Experimental settings

4.1.1. Data

Four fault diagnosis benchmarks are processed and analyzed for evaluation purposes. A detailed description of these benchmarks is given in the following:

1. **Microsoft Azure Predictive Maintenance** [28]: It can be used for both fault diagnosis and predictive maintenance evaluations. It contains several operating conditions with failure and repair archives of a machine. It contains 90,000 samples, each sample is identified by a timestamp.
2. **NASA Milling Dataset** [29]: It contains an analysis of runs of milling machines with several executing constraints. Three kinds of sensors are used including acoustic emission, vibration, and current sensors. Different cases are generated with increasing the number of runs. It contains 167 cases separated by a unique timestamp. From these cases, we generate 50,000 samples. Each of which simulates one case from 167 cases.
3. **Preventive to Predictive Maintenance** [30]: It deals with the practice-relevant degradation process. It is generated from different scenarios that do usually match the normal situation. Several configurations are implemented to achieve better evaluation. It contains 20,480 samples represented by timestamps.
4. **CWRU Bearing Dataset** [31]: It contains time series for motor fault diagnosis. The deficiency is introduced at every single value and measured in inches. Different deficiency diameter sizes are observed, and these deficiencies are located in different parts: ball, inner and outer races. It contains 48,967 samples with different features. The projection on the anomaly class with the timestamp is done as a preprocessing step.

4.1.2. Evaluation metric

To evaluate the proposed framework, the detection rate DR is calculated and used as a metric of comparison. This is widely used to evaluate fault diagnosis algorithms and is defined by:

$$DR = \frac{DF}{AF} \times 100, \tag{7}$$

where DF is the number of detected faults, and AF is the number of all faults observed in the system.

4.1.3. Baseline methods

In the experiments, we compare the ADL-FDI4 framework with the following baseline fault diagnosis solutions:

1. **semi-DCNN (semi-deep Convolution Neural Network)** [32]: It is a hybrid method that combines support vector machine with DCNN for multi-fault diagnosis systems. The multi-features are automatically extracted by the DCNN, the support vector machine then uses these features to decide the machine state.
2. **FD-SAE (Feature Distance Stack AutoEncoder)** [33]: The feature extraction is performed with the autoencoder mechanism by respecting diversification with computing the distance among the features. The support vector machine uses these features to classify and identify the faults in the system.
3. **GA-SVR (Genetic Algorithm with Support Vector Regression)** [34]: It is a hybrid support vector regression and genetic algorithm for identifying fault diagnosis in the industrial equipment. The genetic algorithm is used to optimize the parameters of the support vector regression.

4.2. Parameters' settings

The first experiment aims to tune the hyper-parameters of the proposed ADL-FDI4 framework. As explained in Section 3.3, Branch-and-Bound plays the role of supervision of the different deep learning models developed in this research work. Parameters' settings are an important preliminary step where the hyper-parameters of the different deep learning models should be carefully analyzed and tuned. The common parameters of the deep learning models used in this work are the number of epochs, the learning rate, and the number of batches. To analyze such hyper-parameters, intensive experiments have been carried out by launching the Branch-and-Bound optimization algorithm for each execution. Thus, the number of epochs is varied from 1 to 100 with 1 step, the learning rate is varied from 0 to 1 with 0.1 step, and the number of batches is varied from 8 to 512 with 8 steps. Branch-and-Bound explores the hyper-parameters space and finds the best parameters for each dataset. The summary of the best parameters' values is illustrated in Table 1. In the next two sections, our model uses the best value of each hyper-parameter and is compared to the state-of-the-art models on these four benchmark datasets in terms of accuracy and running time.

4.3. Accuracy performance

Our initial experiments aimed at evaluating the accuracy of ADL-FDI4 compared to the baseline fault diagnosis solutions: semi-DCNN, FD-SAE, and GA-SVR. Using the four datasets mentioned above. By varying the number of faults as input from 10 to 100, Fig. 3 shows that ADL-FDI4 outperforms the three baseline algorithms in terms of

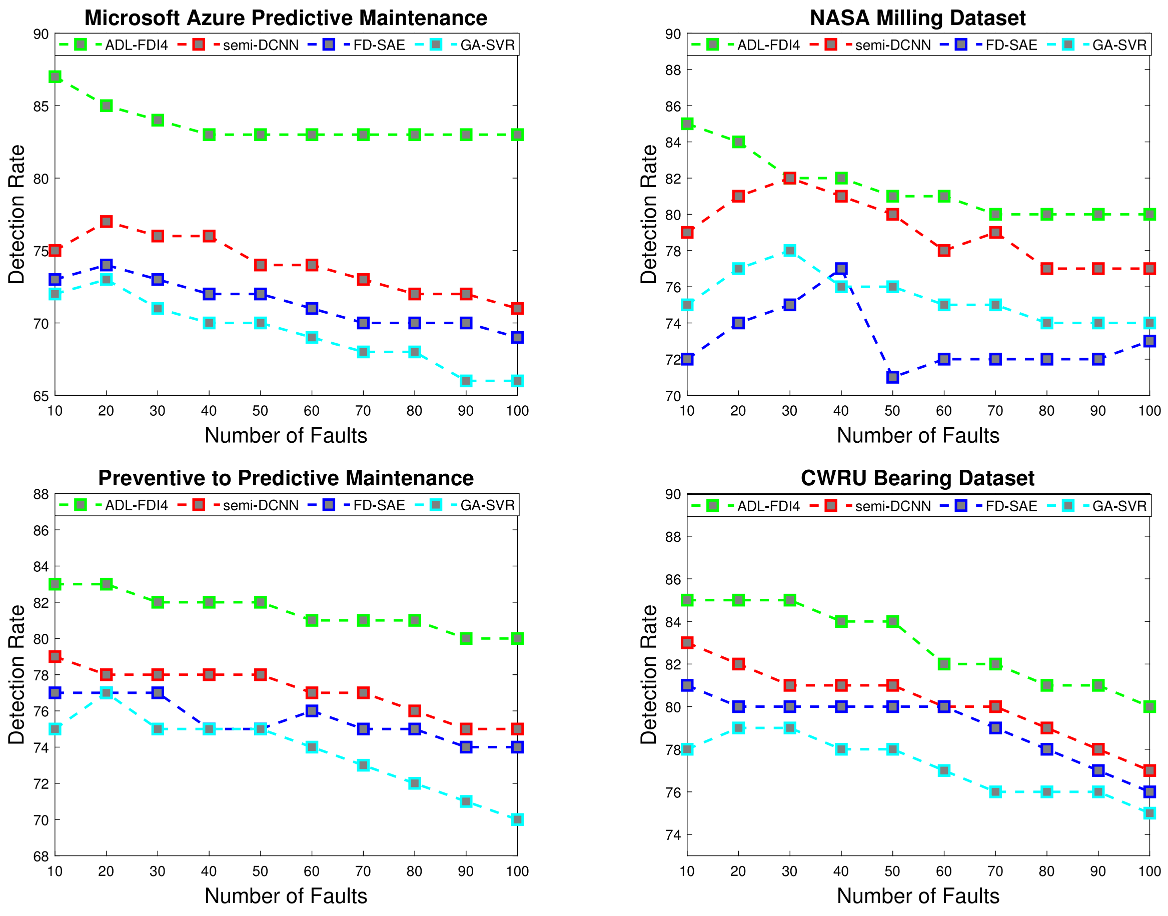

Fig. 3. Accuracy of the ADL-FDI4 compared to the state-of-the-art fault diagnosis solutions.

Table 1
Best parameters of ADL-FDI4.

Dataset	Epochs	Learning rate	Batches
Microsoft Azure Predictive Maintenance	53	0.67	16
NASA Milling	65	0.45	32
Preventive to Predictive Maintenance	73	0.62	8
CWRU Bearing	33	0.86	16

Table 2
Training performance of the ADL-FDI4 and the state-of-the-art fault diagnosis solutions on Big Data.

Dataset X 100	ADL-FDI4		semi-DCNN		FD-SAE		GA-SVR	
	CPU	Acc.	CPU	Acc.	CPU	Acc.	CPU	Acc.
Micro. Azure Pred. Maint.	1543	98	1432	96	896	92	906	89
NASA Mil. Data.	1633	95	1230	93	993	90	1076	88
Prev. Pred. Maint.	1549	99	1260	95	1120	94	998	91
CWRU Bear. Data.	1205	97	1006	96	879	93	976	92

detection rate. The detection rate of the ADL-FDI4 reached 83% for dealing with 100 faults of the Microsoft azure predictive maintenance dataset. Whereas the detection rate for the other models goes under 71% for handling the same scenario. These results are obtained thanks to the efficient combination between deep learning and the Branch-and-Bound strategy for fault diagnosis. The Branch-and-Bound strategy can efficiently tune the hyper-parameters of the different deep learning models used in the ADL-FDI4.

4.4. Runtime performance

The second experiments aim at evaluating the runtime of the ADL-FDI4 compared to the baseline fault diagnosis solutions: semi-DCNN, FD-SAE, and GA-SVR using the four datasets mentioned above. By varying the number of faults as input from 10 to 100, Fig. 4 shows that ADL-FDI4 outperforms the three baseline models in terms of runtime. However, the difference in performance for the three models is too small for the Microsoft azure predictive maintenance dataset and it is high for the other datasets. The difference in runtime between ADL-FDI4 and the baseline algorithms does not exceed 10 milliseconds for dealing with 100 faults of the Microsoft azure predictive maintenance dataset. Whereas the difference in runtime between ADL-FDI4 and the baseline algorithms reaches 25 milliseconds for handling the same scenario on the CWRU Bearing dataset. These results are explained by the fact that the other algorithms are complex methods that combine both the deep learning architectures for extracting the features and the traditional machine learning algorithms for the fault diagnosis process.

4.5. Training performance on big data

The last experiment is meant to show the ability to train the proposed framework on Big Data. Table 2 presents the training performance (the runtime in seconds, and the accuracy in percentage) of the proposed solution and the state-of-the-art solutions for handling 100 times of the data described above. From these results, we can say that the proposed framework gets a better accuracy than the other solutions, whatever is the data used. For instance, it reaches 99% of accuracy with 100 times the Preventive to Predictive Maintenance dataset, while the other solutions do not exceed 95% for the same data and with the same data size. This result is explained by the fact that the proposed framework is more robust, by the fact that it contains three different deep learning architectures. In addition, the efficient selection of the

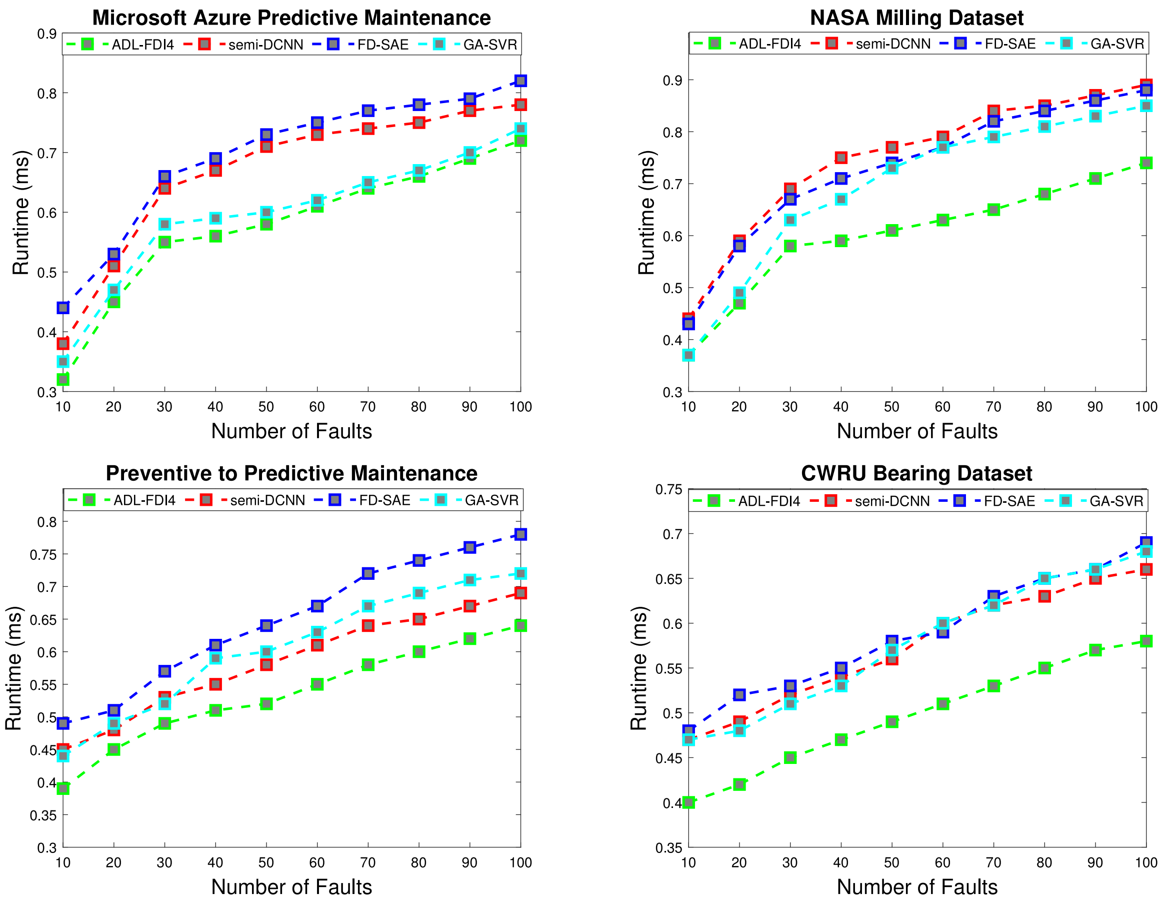

Fig. 4. Runtime of the ADL-FDI4 compared to the state-of-the-art fault diagnosis solutions.

hyper-parameters with the Branch-and-Bound allows for the increase of the training accuracy performance. However, the proposed framework needs more time to train compared to the other solutions. This is explained by the fact that three architectures are used instead of one, and also by the time required for the hyper-parameter optimization process.

4.6. Discussion

This section shows the lessons learned from the investigation of the ADL-FDI4 framework to solve the issue of sustainable fault diagnosis in (B5G)/6G Industry 4.0 applications. The first finding of this study is that the proposed framework is generic and can efficiently deal with heterogeneous data, using a combination of different deep learning architectures. Another finding is that the deep learning models benefit from the Branch-and-Bound strategy for hyper-parameters tuning. The enumeration search tree is first created, and then an efficient heuristic is used to prune the irrelevant branches. We also show that ADL-FDI4 outperforms the state-of-the-art algorithms both in terms of detection rate and running time, in addition to handling heterogeneous data, which makes it a more sustainable approach.

5. Conclusion

In this paper, we proposed an Advanced Deep Learning framework for Fault Diagnosis in (B5G)/6G Industry 4.0 applications (ADL-FDI4). This framework addresses two issues in the current technologies for fault diagnosis. The first issue is the non-ability of the state-of-the-art algorithms to handle heterogeneous data, meaning that more than one Fault Detection computational device has to be used to process all the data formats, in addition to not exploiting all the information available that can help increase the accuracy of the model. And this raises sustainability concerns. The second issue is that these algorithms rely on inefficient exhaustive search hyper-parameters tuning strategies. ADL-FDI4 aims at addressing the first issue by benefiting from the combination of different deep learning architectures, which are LSTM for handling time series, CNN for image processing, and graph CNN for graph data. And solving the second issue by relying on a smart Branch-and-Bound strategy to explore the parameters' space and optimize the parameters' choice. For performance evaluation, our model is compared to three state-of-the-art baseline fault diagnosis solutions using four diagnosis benchmarks. The experimental results show that ADL-FDI4 outperforms the state-of-the-art solutions in terms of detection rate and running time, and for that, it consumes less energy. In addition to handling heterogeneous data, which implies that one computational fault detection device is sufficient to handle all data formats. All of this makes ADL-FDI4 a more sustainable fault detection approach.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All authors approved the version of the manuscript to be published.

References

[1] Y. Guo, Z. Zhao, K. He, S. Lai, J. Xia, L. Fan, Efficient and flexible management for industrial internet of things: A federated learning approach, Comput. Netw. 192 (2021) 108122.

[2] O. Gungor, T.S. Rosing, B. Aksanli, DOWELL: diversity-induced optimally weighted ensemble learner for predictive maintenance of industrial internet of things devices, IEEE Internet Things J. (2021).
[3] K. Tange, M. De Donno, X. Fafoutis, N. Dragoni, A systematic survey of industrial internet of things security: requirements and fog computing opportunities, IEEE Commun. Surv. Tutor. 22 (4) (2020) 2489–2520.
[4] J.C.W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, M. Aloqaily, Privacy-preserving multiobjective sanitization model in 6G IoT environments, IEEE Internet Things J. 8 (7) (2021) 5340–5349.
[5] C.F. Cheng, Y.C. Chen, J.C.W. Lin, A carrier-based sensor deployment algorithm for perception layer in the iot architecture, IEEE Sens. J. 20 (17) (2020) 10295–10305.
[6] W.U. Khan, M.A. Javed, T.N. Nguyen, S. Khan, B.M. Elhalawany, Energy-efficient resource allocation for 6G backscatter-enabled NOMA IoV networks, IEEE Trans. Intell. Transp. Syst. (2021).
[7] T.G. Nguyen, T.V. Phan, D.T. Hoang, T.N. Nguyen, C. So-In, Federated deep reinforcement learning for traffic monitoring in sdn-based iot networks, IEEE Trans. Cogn. Commun. Netw. (2021).
[8] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Bus. Inf. Syst. Eng. 6 (4) (2014) 239–242.
[9] A.G. Frank, L.S. Dalenogare, N.F. Ayala, Industry 4.0 technologies: implementation patterns in manufacturing companies, Int. J. Prod. Econ. 210 (2019) 15–26.
[10] G. Aceto, V. Persico, A. Pescapé, A survey on information and communication technologies for industry 4.0: state-of-the-art, taxonomies, perspectives, and challenges, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3467–3501.
[11] B. Jeon, J.-S. Yoon, J. Um, S.-H. Suh, The architecture development of industry 4.0 compliant smart machine tool system (SMTS), J. Intell. Manuf. 31 (8) (2020) 1837–1859.
[12] B. He, K.-J. Bai, Digital twin-based sustainable intelligent manufacturing: A review, Adv. Manuf. 9 (1) (2021) 1–21.
[13] K.S. Kiangala, Z. Wang, An effective predictive maintenance framework for conveyor motors using dual time-series imaging and convolutional neural network in an industry 4.0 environment, IEEE Access 8 (2020) 121033–121049.
[14] C.N.E. Kan, R.J. Povinelli, D.H. Ye, Enhancing multi-channel eeg classification with gramian temporal generative adversarial networks, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2021, pp. 1260–1264.
[15] J. Dalzochio, R. Kunst, E. Pignaton, A. Binotto, S. Sanyal, J. Favilla, J. Barbosa, Machine learning and reasoning for predictive maintenance in industry 4.0: current status and challenges, Comput. Ind. 123 (2020) 103298.
[16] L. Silvestri, A. Forcina, V. Introna, A. Santolamazza, V. Cesarotti, Maintenance transformation through industry 4.0 technologies: A systematic literature review, Comput. Ind. 123 (2020) 103335.
[17] T. Küfner, S. Schönig, R. Jasinski, A. Ermer, Vertical data continuity with lean edge analytics for industry 4.0 production, Comput. Ind. 125 (2021) 103389.
[18] L. Kaupp, H. Webert, K. Nazemi, B. Humm, S. Simons, CONTEXT: An industry 4.0 dataset of contextual faults in a smart factory, Procedia Comput. Sci. 180 (2021) 492–501.
[19] B. Natesha, R.M.R. Guddeti, Fog-based intelligent machine malfunction monitoring system for industry 4.0, IEEE Trans. Ind. Inform. (2021).
[20] M.A. Rahman, M.S. Abuludin, L.X. Yuan, M.S. Islam, A.T. Asyhari, Educhain: cia-compliant block-chain for intelligent cyber defense of microservices in education industry 4.0, IEEE Trans. Ind. Inform. (2021).
[21] Z. Long, X. Zhang, M. He, S. Huang, G. Qin, D. Song, Y. Tang, G. Wu, W. Liang, H. Shao, Motor fault diagnosis based on scale invariant image features, IEEE Trans. Ind. Inform. (2021).
[22] Y. Liu, W. Yu, T.S. Dillon, W. Rahayu, M. Li, Empowering iot predictive maintenance solutions with AI: A distributed system for manufacturing plant-wide monitoring, IEEE Trans. Ind. Inform. (2021).
[23] Y. Yu, L. Guo, H. Gao, Y. Liu, T. Feng, Pareto-optimal adaptive loss residual shrinkage network for imbalanced classification of machinery fault diagnostics, IEEE Trans. Ind. Inform. (2021).
[24] A. Hazra, M. Adhikari, T. Amgoth, S.N. Srirama, Intelligent service deployment policy for next-generation industrial edge networks, IEEE Trans. Netw. Sci. Eng. (2021).
[25] A. Hazra, M. Adhikari, T. Amgoth, S.N. Srirama, Collaborative AI-enabled intelligent partial service provisioning in green industrial fog networks, IEEE Internet Things J. (2021).
[26] M. Adhikari, A. Munusamy, A. Hazra, V.G. Menon, V. Anavangot, D. Puthal, Security and privacy in edge-centric intelligent internet of vehicles: issues and remedies, IEEE Consumer Electron. Mag. (2021).
[27] M. Adhikari, A. Munusamy, N. Kumar, S.N. Srirama, Cybertwin-driven resource provisioning for ioe applications at 6g-enabled edge networks, IEEE Trans. Ind. Inform. (2021).
[28] P. Strauß, M. Schmitz, R. Wöstmann, J. Deuse, Enabling of predictive maintenance in the brownfield through low-cost sensors, an iiot-architecture and machine learning, in: 2018 IEEE International conference on big data (big data), 2018, pp. 1474–1483.
[29] A. Agogino, K. Goebel, Mill data set. best lab, uc berkeley. nasa ames prognostics data repository, 2007.
[30] S. Hagmeyer, F. Mauthe, M. Dutt, P. Zeiler, Preventive to predictive maintenance dataset, 2021.
[31] L. Eren, T. Ince, S. Kiranyaz, A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier, J. Signal Process. Syst. 91 (2) (2019) 179–189.
[32] Y. Xue, D. Dou, J. Yang, Multi-fault diagnosis of rotating machinery based on deep convolution neural network and support vector machine, Measurement 156 (2020) 107571.
[33] M. Cui, Y. Wang, X. Lin, M. Zhong, Fault diagnosis of rolling bearings based on an improved stack autoencoder and support vector machine, IEEE Sens. J. 21 (4) (2020) 4927–4937.
[34] W. Jingjing, L. Qinming, Y. Chunming, L. Guanlin, Fault diagnosis of mechanical equipment based on GA-SVR with missing data in small samples, J. Syst. Simul. 33 (6) (2021) 1342.