Fault Detection and Isolation in Industrial Processes Using Deep Learning Approaches

Rahat Iqbal, Tomasz Maniak, Faiyaz Doctor, and Charalampos Karyotis

***Abstract*—Automated fault detection is an important part of a quality control system. It has the potential to increase the overall quality of monitored products and processes. The fault detection of automotive instrument cluster systems in computer-based manufacturing assembly lines is currently limited to simple boundary checking. The analysis of more complex nonlinear signals is performed manually by trained operators, whose knowledge is used to supervise quality checking and manual detection of faults. We present a novel approach for automated Fault Detection and Isolation (FDI) based on deep learning. The approach was tested on data generated by computer-based manufacturing systems equipped with local and remote sensing devices. The results show that the approach models the different spatial/temporal patterns found in the data. The approach can successfully diagnose and locate multiple classes of faults under real-time working conditions. The proposed method is shown to outperform other established FDI methods.**

***Index Terms*—Artificial Neural Networks (ANNs), computer-aided manufacturing, deep learning, fault detection, machine learning, manufacturing automation.**

I. Introduction

THE development of fault detection systems for complex real-world industrial processes is difficult and poses many challenges [1]. Modern computer-based manufacturing systems consist of many manufacturing cells performing a range of assembly operations and functional tests. The cells are controlled by computer software supervising a given production process many of which are custom built [2].

For computers assigned to the supervision of manufacturing plants, one of the most important tasks is to detect and diagnose product faults. The first step in this task is to acquire the data necessary for process analysis. The earliest inspection systems utilized a small number of data generating processes and sensing elements. This resulted in only a limited amount of data which could be analyzed by engineers for the fault identification process; a more methodical approach supported by structured data analysis was lacking.

To this day, the only forms of fault detection used in many manufacturing plants are those based on limit checking [3]. In such a case, minimal and maximal values, called thresholds, are specified for a given characteristic in the manufacturing process for a product. A normal operational state is when the value of a feature is within these specified limits. Although simple, robust, and reliable, this method is slow to react to changes of a given characteristic of the data and fails to identify complex failures, which can only be identified by looking at the correlations between features. Another problem with this approach is the challenge of specifying the threshold values for a given characteristic [4].

To resolve the above problem, most manufacturing companies have historically adopted a technique called Statistical Process Control (SPC) that was developed in 1920s by Walter Shewhart. SPC is a set of different methods to understand, monitor, and improve process performance over time [5]. The most apparent limitation of SPC methods is the fact that they are concerned mainly with one input at a certain point in time [6] and ignore the spatial/temporal correlation which could otherwise help to detect and isolate potential faults. It is therefore crucial to investigate and propose new fault detection and isolation techniques based on more sophisticated modeling capabilities of methods, such as advanced intelligent data analysis and machine learning approaches.

Modern computer-based manufacturing systems produce a large volume of data generated by sensor and control signals during the manufacturing process. The data contain valuable information about the state of the system and its potential faults. In such systems, the available automated solutions to assist engineers with fault detection are limited and only consider one measured characteristic of a manufacturing process at a time. This creates a simplified static image of a complex dynamic system. State-of-the-art tools can consider multiple characteristics but disregard the temporal aspect of the signal, creating a static model of the system. More significantly, these tools ignore various correlations between multiple characteristics, which dynamically change over time and provide additional information about a fault occurrence. Another problem is the limited automation of the fault classification and inference, making it necessary to train staff/engineers to use the tools effectively. This results in additional cost and places constraints on the flexible use of

Manuscript received February 4, 2019; accepted February 23, 2019. Date of publication February 28, 2019; date of current version May 2, 2019. Paper no. TII-19-0392. *(Corresponding author: Rahat Iqbal.)*

R. Iqbal is with the Institute of Future Transport and Cities, Coventry University, Coventry CV1 5FB, U.K. (e-mail: r.iqbal@coventry.ac.uk).

T. Maniak was with Nippon Seiki, Redditch B98 9NR, U.K. He is currently with Interactive Coventry, Ltd., Coventry CV1 2TT, U.K. (e-mail: tomasz.maniak@interactivecoventry.com).

F. Doctor is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: fdocto@essex.ac.uk).

C. Karyotis is with Interactive Coventry, Ltd., Coventry CV1 2TT, U.K. (e-mail: charalampos.karyotis@interactivecoventry.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2902274

human resources. Likewise, these methods cannot detect faults at an early stage, respond to constantly changing fault sources or learn new fault types from multitype spatial-temporal production data. Ignoring the aforementioned problems leads to extensive production downtime and waste of resources, unsafe machinery, poor production yield, and suboptimal human resource allocation.

The rest of this paper is organized as follows. Section II provides an overview of existing FDI methods used in the manufacturing environment. Section III discusses the proposed approach. Section IV discusses the implementation and Section V describes the evaluation of the proposed approach in a real-world setting. Finally, in Section VI, conclusions and future work are discussed.

II. Existing Fault Detection Methods

The importance of using FDI has been first recognized in safety critical areas such as flight control, railways, medicine, nuclear plants, and many more. The need for fault detection is also more relevant nowadays due to the new application of computational intelligence for data analysis performed by real-time systems. This is especially true in real-time energy efficient management of distributed resources [7], real-time control, and mobile crowdsensing [8] (both a vital part of smart and connected communities) and the protection of sensitive information collected by wearable sensors [9].

A conventional method for ensuring the fault-free operation of manufacturing production lines is to periodically check the process variables, which include software configuration validation, sensor validation, measurement device calibration, and preventive maintenance [10]. This method is widely popularized in industry and used for preventing and detecting abrupt failures. However, it is not able to detect failures that can only be detected by continuous assessment of variables, such as incipient process faults, which are especially relevant in the manufacture of microelectronic components. Owing to an increase in the process complexity and sophistication of production equipment, this method is no longer cost effective and impractical to implement on large-scale computer-based production lines [11].

Fault detection methods can be mostly categorized into two main groups: hardware redundancy and analytical redundancy [12]. The main idea behind redundancy-based methods is to generate a residual signal which represents a difference between the normal behavior of a system and its actual measured behavior. By considering this comparison, a fault occurrence can be detected. Hardware redundancy is based on creating the residual signal by using hardware [13]. The general idea behind this approach is to measure a given process variable with more than one sensor and detect a fault by performing consistency checks on the different sensors.

Analytical redundancy is based on creating the residual signal from a mathematical model which can be developed by analyzing either the actual measurements, or the underlying physics of the process. There are three main approaches to analytical redundancy: model-based methods, data-driven methods, and knowledge-based expert systems [14]. They are all categorized based on *a priori* knowledge, which is required for the model. Model-based methods require a good mathematical model of the monitored system which can be acquired using parameter estimators, parity relations, or state observers such as Luenberger observers and Kalman filters [12]. Data-driven methods, instead of creating a mathematical model, use historical data recorded by sensors to monitor a given system. The data are used to describe and model the normal behavior of that system, which is subsequently used to generate a residual signal. The data-driven methods can be used only if the given system can generate enough data from the sensors [15]. Finally, a knowledge-based expert system uses domain knowledge which is very often described as a set of rules [16].

A different approach for the classification of fault detection methods is to consider the different methods from the perspective of the variables that are used to detect a fault [17]. In this context, methods based on analyzing single signals or multiple signals and models can be considered. The single signal methods consider one process variable in isolation from other variables. They include methods based on limit and trend checking such as fixed threshold, adaptive threshold, or change detection methods [17]. Thresholds are set to detect whether a given characteristic of the system falls outside the acceptable minimal and maximal values. This method, while simple and reliable, is slow to react to changes in the value of a characteristic over time and is incapable of identifying complex failures. To overcome this problem, a set of methods used to analyze multiple signals have been used. Those are: principle component analysis, parameter estimators, Artificial Neural Networks (ANNs), state observers, parity equations, and state estimators [15]. These methods identify faults by analyzing the correlations between multiple system variables. Finally, a set of temporal methods for both single and multiple signal variables have been used, which have provided the tools necessary to identify faults in high frequency signals. These methods are necessary for dynamic systems where a fault can only be identified by looking at the way signals change over time. Examples of these methods are: spectrum analysis, wavelet analysis, and analysis of correlations [18].

Many fault detection systems used in computer-based manufacturing environments are rule-based expert systems. An expert system is a specialized system that solves problems in a domain of expertise. Such systems simulate human reasoning for a problem domain, perform reasoning over a set of previously defined logical statements, and then solve the problem using heuristic knowledge [19]. An expert system is a computer program consisting of a large database of if-then-else rules which mimics the cognitive behavior and knowledge of human experts [33]. The main advantages of developing such systems include: ease of implementation and development, ease of fault interpretation, transparent logical reasoning, and the ability to deal with noise and uncertainty in the data. Because of the large variety of processes to which expert systems are applied, there is a significant number of papers and scientific literature devoted to their implementation [20]–[22]. Expert systems require significant human effort and experience to precisely describe the heuristic knowledge of a monitored process. Another limitation in using this method is that the database of symptoms should be

modified each time a new rule is added. Finally, another problem is their rigid structure as they lack the ability to fully express the real-world understanding of the underlying process [23]. This is the reason why they fail to generalize and adapt when a new condition is encountered that is not explicitly defined in the knowledge base. This kind of knowledge is called "shallow" since it lacks the deep understanding of the underlying physics of the system [23]. That is why expert systems are very often impractical for systems that have many variables, or systems with significant variability.

Each manufacturing process is subject to uncertainty and random disturbances. This uncertainty comes from many sources, including measurement uncertainty, human performance, or part variation. That is why sometimes a problem of fault detection needs to be formulated in the context of stochastic systems. These systems are defined using a probability distribution, which corresponds to the state of the system under normal working conditions. Any change in that probability distribution can be an indicator of a fault occurrence in the monitored system. In real-time systems, observations are analyzed sequentially, and fault occurrence is identified based on the observations over a particular time period [20]. By monitoring the variable and considering it as a function of time, a fault occurrence could be identified and a corrective action introduced. This action would return the system to its normal operation by resetting the variable to its desired value. Although SPC charts are still widely used in manufacturing process control, the charting methods have not kept up with the progress in data acquisition. Another problem with SPC analysis is the fact that it is slow to respond to subtle changes in monitored variables. Finally, SPC charts are generally concerned with the input of one variable in isolation; therefore, if a given variable is dependent on other variables, then the charts can be misleading.

III. Proposed Approach

To address the problem of FDI, we have proposed a novel universal biologically inspired generative-modeling approach as shown in Fig. 1. The approach is designed to mimic the natural fault detection functions that have evolved and developed in the mammalian brain and is inspired by a theory proposed by Jeff Hawkins [24].

The proposed approach is capable of modeling complex correlations between input values and the temporal consequences between different input states of the system (phrased in this paper as spatial–temporal correlations) in high volumes of data. Consequently, the approach predicts the future states of a system based on its previous behavior while taking into account significant noise in the data. The approach can automatically learn complex real-world patterns to identify abnormal conditions. This gives it a competitive advantage over rival methods where substantive human supervision is required. Due to its unique capability for handling data invariances, the approach is able to process a broad range of data types to discover patterns, which are too complex for humans or standard machine learning techniques to identify.

The main elements of the proposed approach are as follows; see Fig. 1. Initially data produced from several hardware/software sources (data layer) are transformed into individual signals. Those signals (input layer) comprise of various data types and represent a measured physical characteristic of a monitored process. Depending on the type of signal, they are encoded in one of the following ways. This encoding is performed in the data transformation layer as follows: for signals representing a categorical entity, the values are encoded using one-hot encoding i.e., the input space $M_i = \mathbb{Z}_k$ is mapped to k binary features encoding that input. Where a signal is continuous, a range of that signal is considered and divided into a fixed number of bins depending on the mean and standard deviation of the signal. The input space is then mapped to k-binary features encoding the bin that the value falls into. Finally, binary signals are copied without the need to use a dedicated encoder. During an operation of the manufacturing system, at each time t, the measured physical characteristics $\{ p_0, p_1, \ldots, p_n \} \in P$ (where P is a set of all measured physical characteristics for a given manufacturing system) are encoded and concatenated to create a sparse binary input vector. The input vectors $x_i \in \beta$ where β s a set of all possible input vectors and $x_i \in [0, 1]^d$ are generated during that operation which changes dynamically over time and creates a sequence of input vectors $S = (x_t)_{t=0}^{n}$. Here, d denotes the elements in an input vector x_i and the number of measured physical characteristics is n. For typical complex manufacturing systems, the following is true $n > 150$ and depending on the type of the physical characteristic, the number of elements $d > 1000$. A problem with the current representation of x_i is that although the individual elements of that vector are correlated there is no mechanism which would capture those correlations. To solve the above problem, the method uses a set of Deep Autoencoders (DAEs) [25] to learn a vector space embedding $v_i \in A$ where $A \in \mathbb{R}^l$. By using an autoencoder, a mapping $e : \beta \mapsto A$ is achieved which represents x_i in a continuous vector space where correlated input vectors are mapped to nearby points.

The discovery of correlations between individual inputs is determined by the spatial transformation of input space into a transformed vector-space embedding, by using the feature encoder. The continuous space of vector embedding cannot be directly used to infer a current state of a monitored system, instead hierarchical clustering is performed on the transformed features derived from the DAE, to extract the possible states for the modeled system. The process of mapping input space into vector-space embedding and performing hierarchical clustering using the distance between individual input vectors is referred to as spatial pooling. The main purpose of this operation is to reduce the input space to a fixed number of the most probable states of the underlying system being modeled. Temporal sequence learning is used to train the model on the different temporal-consequential relations between probable states of the system. This is used to infer the next predicted state of the inputs as compared to the actual behavior of the system, which is termed as temporal inference. The spatial pooling and temporal-inference elements of the approach combine to produce a spatial–temporal model of the operational behavior of the system being modeled.

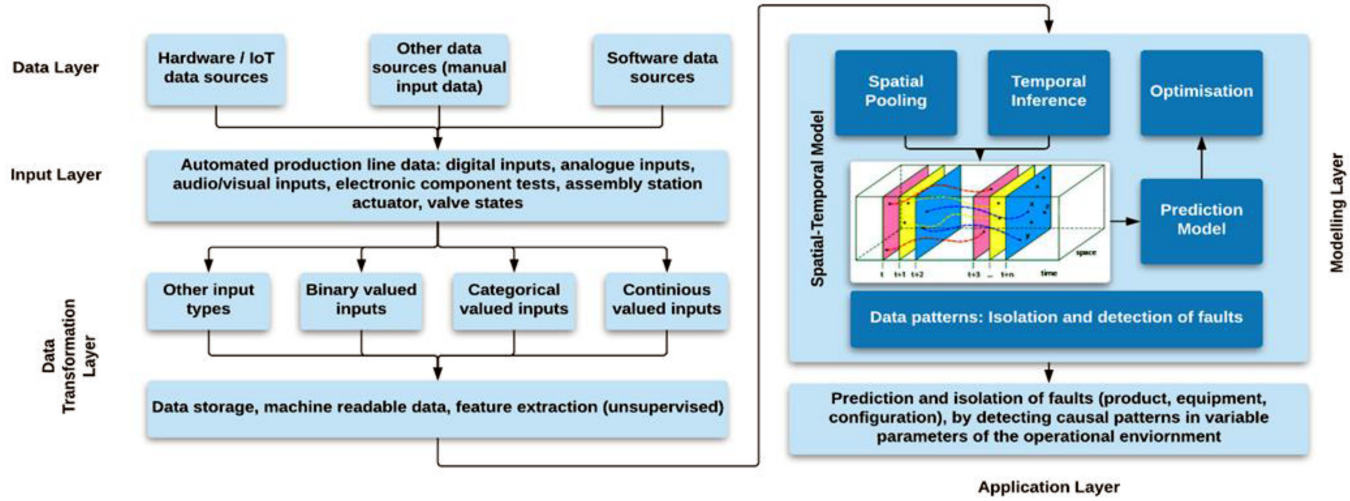

Fig. 1. Proposed approach.

The model can then be used in combination with prediction and classification approaches such as standard ANNs, to predict future behavior of the system under different operational conditions and detect deviations and changes in behavior that might signify an underlying unknown effect or problem. The prediction model can further provide inputs to the optimization framework or an interpretable fuzzy decision model that is able to optimize processes based on quantitative and qualitative inputs from various sources. This approach can therefore be used to determine behavior changes and deviations of complex systems. The output of the model is transferred for further control of the manufacturing production system; see application layer part of Fig. 1.

IV. Implementation

The approach has been implemented using the Python programming language. The implementation of the proposed approach makes use of the Theano library which benefits from dynamic C code generation, stable and fast optimization algorithms, as well as integration with the mathematical NumPy [29] library.

The implementation is divided into a learning module and a real-time module. The learning module performs continuous learning of the parameters for both spatial pooling and temporal inference and uploads them into a database, which is shared with the real-time module. The real-time module performs real-time FDI with the use of parameters stored in the shared database. The module does not perform any learning and is concerned only with the execution of the model with previously learned parameters. This operation of splitting the learning process from the actual execution process is necessary to ensure real-time operation which would otherwise be unattainable. The execution of the learning module is performed on a dedicated server, with the deployed module running as a service. Initially, the module acquires several data samples from an SPC database. The database contains a log of all signals generated by the execution of the manufacturing process as they unfold in time. These data are stored in a database as textual information and loaded by the learning module to computer memory as a list of string objects. Each element of this list represents the current values for all manufacturing signals for a given time frame f_i. The elements in that list are first fragmented into separate signals and based on their type individually encoded into sparsely distributed representations (SDR). The SDR encodings for each signal at time t are combined into a binary array to create an input vector. This process is repeated for the remaining elements of that list and results in a new list of binary input vectors being created and subsequently used as an input to the DAE. An optimization algorithm is executed to adjust parameters of the DAE model, thus minimizing the error on the input reconstructions. The learned parameters of the model are saved and reused during the next iteration of the algorithm.

The data generated by the DAE are subsequently processed by the hierarchical clustering module, which extracts meaningful information about the data structure of the feature space. The dendrogram created by the hierarchical clustering process is cut at a certain height to partition the feature space into multiple regions. For each region, a centroid is assigned and saved to a dictionary. This dictionary is used to map signals for each time frame f_i into a state s_i where $s \in \mathbb{N}$. The output of this operation creates a list of temporal transitions between the different states. The list can therefore be considered to describe state representations of an underlying Markov process. The transition probabilities between the different states s_i are discovered and used to populate the transition matrix of an n-order Markov model. To reduce the memory requirements necessary to store the transition matrix, it is implemented as a dictionary. The entries of the dictionary are saved in the database and used by the real-time module to predict future states of the monitored system. This operation concludes the first iteration of the algorithm. The entire process is repeated and reinitialized with an acquisition of a new set of data samples from the SPC database. This process is presented in Fig. 2.

The real-time module is integrated with custom-built Industrial Test, Control and Calibration (ITCC) software. It starts its execution by downloading the DAE, centroid dictionary, and

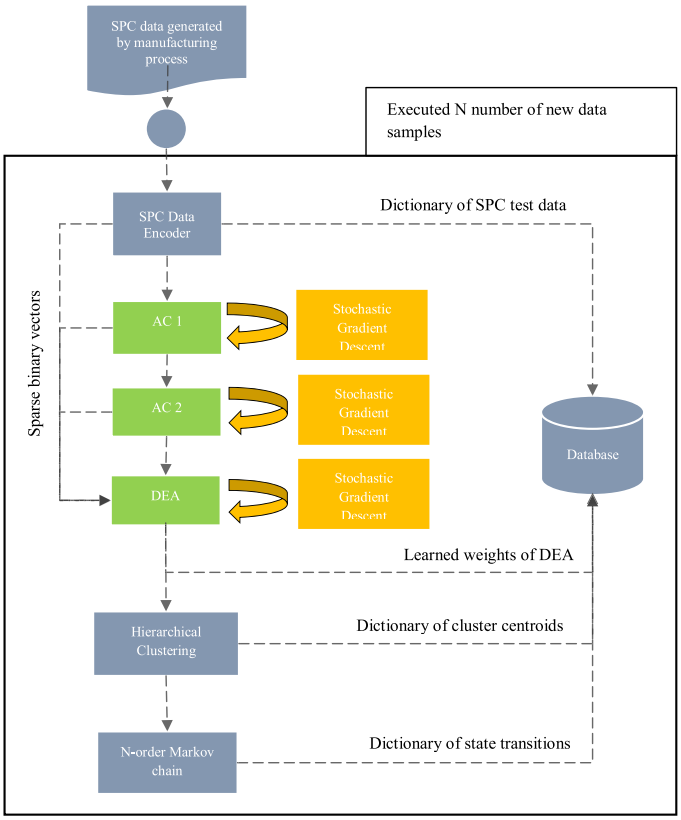

Fig. 2. Learning module execution diagram.

Algorithm 1: Real-Time Module Operation.

```
Load model parameters from database.
Initialise i = 1, MinSmp = 5, LS = []
while(true) do
    j = Acquire input signals
    v = Encode signals j into SDR
    f = Map v into feature space
    s = Extract state identifier for f
    LS[i] = s      LS = list of previous states
    if(i< MinSmp) then    MinSmp = min. samples for inference
        continue
    else
        p = Get state prediction based on LS[1:i–1]
        g = Get centroid for the state p
        res = g – f res = residual vector
        good_noGood = Classify res
        if(good_noGood = = True) then
            return NoFault
        else
            return Fault
        end if
    end if
end while
```

transition dictionary parameters from the shared database. The real-time operation of the manufacturing process, generating spatial–temporal signals, is logged and based on the data type of the signal, transformed into the correct SDR representation. The encoded data are subsequently forward propagated through the DAE structure (initialized with the parameters acquired from the shared database). There is no learning performed in the real-time module. The signals are processed by the DEA and as a consequence transformed into a feature space used as input to the centroid dictionary from where state information is acquired. The inference of the state value is based on the shortest distance between the feature vector and a given centroid. The last n states are saved at any given time and used with the transition dictionary of the n—order Markov chain to infer the future state of the system. The predicted state is transformed back to a feature space and saved to the computer memory. During the next iteration, the predicted feature vector is compared with an actual feature vector generated by the manufacturing process. The residual vector generated by this process is used as an input to a previously trained MLP classifier, which indicates a fault occurrence in the system. This process is described in Algorithm 1.

V. Evaluation

An evaluation of the approach has been performed on a real-world computer-based production line used to manufacture car instrument clusters. In this instance, the real-time module has been integrated with ITCC software running on an autocalibration station. The learning module was executed on a dedicated server with direct access to the production SPC database. A Microsoft SQL server database was used to store and retrieve the learned parameters of the model. The following inputs were used: 327 test ids and corresponding test values (each measuring a different physical characteristic) and their test execution, together with 6 analogue and 91 digital signals.

The data used to train the learning module were composed of 15 000 samples, divided between training (70%), validation (15%), and test (15%) datasets. The dataset is composed of the readings from multiple control and inspection devices that are a part of the manufacturing system. The sensory information is saved as a sequence of activities and measurements that are performed on each individual product. Each entry in this sequence is composed of readings from multiple sensors measuring a value of a given characteristic of either the product or manufacturing equipment at a particular moment in time. The number of measured characteristics can vary between the products, depending on the product type.

The division ratios had been chosen based on the experience gained by working on similar problems. The training dataset was used to determine the weights and biases of the proposed generative model. To measure and optimize the performance of the model with out-of-sample data, the parameters were adjusted and tested on a validation dataset. Initially, an error on input reconstructions as a function of several training epochs for the DAE was analyzed (see Fig. 3).

Fig. 3 clearly shows that there exists a point based on the number of epochs for which the model is trained, after which

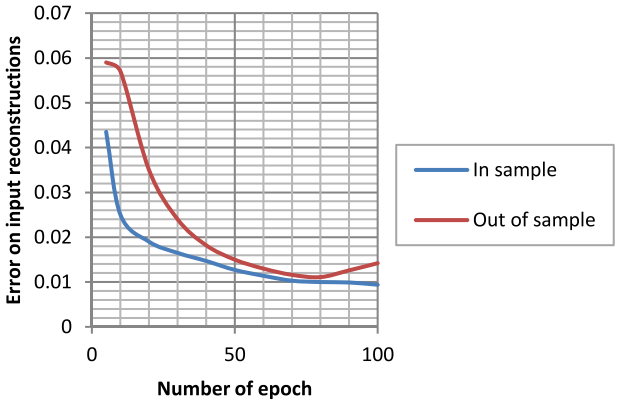

Fig. 3. Error on input reconstructions in sample versus out of sample as a function of epochs trained.

TABLE I
PARAMETERS OF THE MODEL AND THEIR INFLUENCE ON INPUT RECONSTRUCTIONS

Network type		Error rate on input reconstructions
With dropout	Without momentum	0.0069
	With momentum	0.0047
Without dropout	Without momentum	0.0074
	With momentum	0.0071

the reconstruction error for training samples goes down, but the reconstruction error on the validation set increases. This is due to the problem of model overfitting [30] where a model fits the input data too closely and does not generalize well on the out of sample data.

An effective solution to the overfitting problem is the use of a technique called Dropout where the method of setting a random output of a given layer to 0, based on a given probability, was implemented [31]. Extensive experiments have proven the usefulness of such a technique [32]. This technique was applied for this work and was shown to improve the input reconstruction results as presented in Table I. Additionally, an application of a different technique based on learning-rate adaptation called momentum was also considered, where its influence on the error of input reconstruction is measured and presented in Table I.

This study shows a positive influence of the momentum technique on the reduction of error in input reconstruction. Another study was performed to analyze the influence of different optimization methods on the input reconstruction. The results presented in Table II show that the application of the RMSprop achieves the least error rate on input reconstruction.

An important aspect of training Deep Neural Networks (DNNs) is the choice of the network architecture. From this perspective, a number of questions arise. First, how many layers of hidden activation should be considered and second the number of hidden units for each of the network layers. Table III presents the different error rates for input reconstructions for a different number of layers used both with Restricted Boltzmann

TABLE II
SELECTED OPTIMIZATION METHODS AND THEIR PERFORMANCE

Optimisation algorithm used	Error rate on input reconstructions
SGD	0.0047
Adam	0.0042
RMSprop	0.0039
Adadelta	0.0121

TABLE III
ERROR RATE ON INPUT RECONSTRUCTION FOR DIFFERENT NUMBER OF HIDDEN LAYERS

Network		Error rate on input reconstructions
Type	Depth (Architecture)	
Deep Auto-encoder + RBM pre-training	1	0.1436
	3	0.0081
	5	0.0523
Deep Auto-encoder + RWI	1	0.2877
	3	0.1157
	5	0.0691

TABLE IV
ERROR RATE ON INPUT RECONSTRUCTION FOR DIFFERENT NUMBER OF HIDDEN UNITS

Network			Error rate on input reconstructions
Type	Total number of hidden units in the DNN	Maximum execution time (ms)	
Deep Auto-encoder + RBM pre-training	250	164	0.0126
	350	271	0.0074
	450	343	0.0089
	550	527	0.0096
Deep Auto-encoder +RWI	250	164	0.2658
	350	271	0.1043
	450	343	0.3674
	550	527	0.4673

Machine (RBM) pretraining and with Random Weight Initializations (RWI). This study was performed with the use of a validation dataset.

Considering the above, another examination concerning the total number of hidden units in the DNN was performed to test the influence of the number of hidden units used when training the two models with three hidden layers. The maximum execution time and error rates on input reconstructions for real-time operation of the method have been measured and presented in Table IV.

Based on the results from Table IV, an observation can be made that networks consisting of the total of 350 hidden units (across the two layers of DNN 275-75) produced the smallest error on reconstructions. Worth noticing is the fact that as in the previous study as shown in Table III, the same value of parameter works best for both types of the network with pretraining and

TABLE V
FAULT DETECTION PERFORMANCE OF THE PROPOSED METHODS COMPARED WITH RIVAL FDI TECHNIQUES

	Proposed method	HTM+RBM	Template based method	Rule-based method	Bayesian based method
Faults in product	84%	81%	61%	48%	72%
Faults in equipment	72%	68%	59%	51%	57%
Configuration fault	59%	46%	N/A	36%	N/A

RWI. In both cases, the best results are achieved with model pretraining.

Once the best parameters and the architecture of the model were selected, a test dataset was used with an MLP classifier to identify the fault detection accuracy in a real-time setting. The limited number of samples containing a fault and the small variety of different fault types in the dataset required bootstrapping of the acquired production data, with an additional 40% of samples based on simulated fault conditions. To perform more sophisticated fault isolation and identification, the MLP classifier was trained with four-unit *SoftMax* output layers each of them corresponding to one of the following classes: no fault, faults in a product, faults in inspection equipment and configuration faults. The performance of the proposed method was analyzed and compared with rival methods previously applied to FDI, namely, template-based, rule-based and Bayesian-based methods, as shown in Table V. We additionally compared our approach to another state-of-the-art spatial-temporal modeling approach that is a combination of hierarchical temporal memory (HTM) and RBMs. Table V presents the percentage of correctly classified faults in each of the three fault categories, where a performance comparison of the proposed hybrid model with the other applied methods has been shown to assess its effectiveness.

The results confirm that the proposed method is able to achieve a significant increase in fault detection accuracy for all fault types when compared with other FDI methods.

VI. CONCLUSION

While the impact of quality assurance and fault detection on modern industrial processes is widely acknowledged, Big data and machine learning research for industrial automation is not widely popularized. This paper presented a deep learning-based approach for FDI that is capable of processing the richness and high volumes of manufacturing data. The proposed method is capable of modeling multitype spatial–temporal production data with high accuracy. Consequently, the method is characterized with early fault detection, good adaptation to frequent changes in fault sources and automatic identification of new fault types. The proposed algorithm benefits from minimal human process-supervision requirements due to its generative nature. Results prove that the method is able to outperform rival FDI schemes both in accuracy and the range of faults it can detect. The data processing and analytical mechanisms of the proposed method are generic and not limited to fault detection. They can be used in other contexts—for example, to improve analytical capabilities of important enterprise applications [34].

Future areas of research could include the application of parameter-optimization techniques such as genetic algorithm and particle-swarm optimization to improve the modeling capability of the proposed method. Further improvements to the encoder and classifier of the method will benefit the overall performance of the model. An interesting alternative for future research work would be the investigation into the use of recurrent neural networks to improve temporal predictions of the proposed model, especially through the use of long-/short-term memory units. While the focus of this research is a development of robust FDI systems for industrial automation, the researchers also plan to test the proposed system under a usability prism and apply a user-centered design [33] in order to deliver a user-friendly fault detection system that meets industry requirements and user needs.

REFERENCES

[1] X. Dai and Z. Gao, "From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2226–2238, Nov. 2013.
[2] T. Maniak, C. Jayne, R. Iqbal, and F. Doctor, "Automated intelligent system for sound signalling device quality assurance," *Inf. Sci.*, vol. 294, pp. 600–611, 2015.
[3] D. Ferguson, *Removing the Barriers to Efficient Manufacturing Real-World Applications of Lean Productivity*. Boca Raton, FL, USA: CRC Press, 2013, pp. 194–200.
[4] W. H. Woodall and D. C. Montgomery, "Research issues and ideas in statistical process control," *J. Qual. Technol.*, vol. 31, pp. 376–386, 1999.
[5] W. H. Woodall, "Controversies and contradictions in statistical process control," *J. Qual. Technol.*, vol. 32, pp. 341–378, 2000.
[6] C. J. Spanos, H. Guo, A. Miller, and J. Levine-Parrill, "Real-time statistical process control using tool data (semiconductor manufacturing)," *IEEE Trans. Semicond. Manuf.*, vol. 5, no. 4, pp. 308–318, Nov. 1992.
[7] F. F. Qureshi, R. Iqbal, and M. N. Asghar, "Energy efficient wireless communication technique based on cognitive radio for internet of things," *J. Netw. Comput. Appl.*, vol. 89, no. 1, pp. 14–25, 2017.
[8] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of things and big data analytics for smart and connected communities," *IEEE Access*, vol. 4, pp. 766–773, Feb. 2016.
[9] C. Lin *et al.*, "Differential privacy preserving in big data analytics for connected health," *J. Med. Syst.*, vol. 40, no. 4, pp. 1–9, 2016.
[10] J. Hines and D. Garvey, "Process and equipment monitoring methodologies applied to sensor calibration monitoring," *Qual. Rel. Eng. Int.*, vol. 23, no. 1, pp. 123–135, 2007.
[11] N. M. Paz and W. Leigh, "Maintenance scheduling: Issues, results and research needs," *Int. J. Oper. Prod. Manage.*, vol. 14, no. 8, pp. 47–69, 1994.
[12] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis," *Comput. Chem. Eng.*, vol. 27, no. 3, pp. 293–311, 2003.
[13] A. Willsky and H. Jones, "A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems," *IEEE Trans. Autom. Control*, vol. AC-21, no. 1, pp. 108–112, Feb. 1976.
[14] X. Dai and Z. Gao, "From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2226–2238, Nov. 2013.
[15] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part III: Process history based methods," *Comput. Chem. Eng.*, vol. 27, no. 3, pp. 327–346, 2003.
[16] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies," *Comput. Chem. Eng.*, vol. 27, no. 3, pp. 313–326, 2003.

[17] G. Dalpiaz, "Advances in condition monitoring of machinery in non-stationary operations," in *Proc. 3rd Int. Conf. Condition Monit. Mach. Non-Stationary Oper.*, Berlin, Germany, 2013, pp. 137–147.
[18] M. El Hachemi Benbouzid, "A review of induction motors signature analysis as a medium for faults detection," *IEEE Trans. Ind. Electron.*, vol. 47, no. 5, pp. 984–993, Oct. 2000.
[19] L. F. Pau, "Survey of expert systems for fault detection, test generation and maintenance," *Expert Syst.*, vol. 3, no. 2, pp. 100–110, Apr. 1986.
[20] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker, "Expert system framework for fault detection and fault tolerance in robotics," *Comput. Elect. Eng.*, vol. 20, no. 5, pp. 421–435, 1994.
[21] H. R. DePold and F. D. Gass, "The application of expert systems and neural networks to gas turbine prognostics and diagnostics," *J. Eng. Gas Turbines Power*, vol. 121, no. 4, pp. 607–612, 1999.
[22] J. C. da Silva, A. Saxena, E. Balaban, and K. Goebel, "A knowledge-based system approach for sensor fault modeling, detection and mitigation," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 10977–10989, 2012.
[23] J. Singla, "The diagnosis of some lung diseases in a prolog expert system," *Int. J. Comput. Appl.*, vol. 78, no. 15, pp. 37–40, 2013.
[24] J. Hawkins, *On Intelligence*. New York, NY, USA: Times Books, 2004.
[25] D. A. Jackson, "Stopping rules in principal components analysis: A comparison of heuristical and statistical approaches," *Ecology*, vol. 74, no. 8, pp. 2204–2214, 1993.
[26] J. Hawkins and D. George, *Hierarchical Temporal Memory: Concepts, Theory, and Terminology*. Redwood City, CA, USA: Numenta, Inc., 2006.
[27] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[28] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 19, 2006, pp. 153–160.
[29] E. Jones et al., "SciPy: Open source scientific tools for Python," Aug. 25, 2016. [Online]. Available: http://www.scipy.org/
[30] D. M. Hawkins, "The problem of overfitting," *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 1, pp. 1–12, 2004.
[31] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, arXiv:1207.0580.
[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[33] R. Iqbal, N. Shah, A. James, and J. Duursma, "ARREST: From work practices to redesign for usability," *Int. J. Expert Syst. Appl.*, vol. 38, no. 2, pp. 1182–1192, 2011.
[34] R. Iqbal, N. Shah, A. James, and T. Cichowicz, "Integration, optimization and usability of enterprise applications," *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1480–1488, 2013.

Rahat Iqbal received the Ph.D. degree in computer science from Coventry University, Coventry, U.K., in 2005.

He is an Associate Professor with the Institute of Future Transport and Cities, Coventry University, and also the CEO of Interactive Coventry, Ltd., Coventry. He has more than 20 years of experience in project management and leadership and delivered many industrial projects funded by Engineering and Physical Sciences Research Council, Technology Strategy Board, European Regional Development Fund, and local industries (e.g., Jaguar Land Rover, Ltd., Trinity Expert Systems, Ltd.). He has authored or coauthored more than 120 papers in peer-reviewed journals and reputable conferences and workshops. His research interests include big data, computer vision, industrial automation, and Internet of Things.

Tomasz Maniak received the Ph.D. degree in computer science from Coventry University, Coventry, U.K., in 2016.

He is Operations Manager with Interactive Coventry (IC), Ltd., Coventry. Prior to joining IC, he was a Senior Software Engineer/Process Engineer for European manufacturing centre of Nippon Seiki, U.K. He has more than 12 years of commercial experience in automation, robotics, vision inspection and machine/deep learning. He has authored or coauthored several papers and patents in the area of artificial intelligence.

Faiyaz Doctor received the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2006.

He is a Lecturer with the School of Computer Science and Electronic Engineering, University of Essex. He has more than 15 years' experience in the design and implementation of intelligence systems for real-world applications with projects funded by Innovate UK, Harvard University, and Newton Fund. He has authored or coauthored more than 60 papers in peer-reviewed international journals, conferences, and workshops. His research interests include computational intelligence emphasizing on type-2 fuzzy systems, deep learning, and hybrid techniques which he has applied to ambient intelligence, affective computing, industrial automation, and biomedical systems.

Charalampos Karyotis received the Ph.D. degree in computer science from Coventry University, Coventry, U.K., in 2017.

He is the Head of research with Interactive Coventry, Coventry. He has several years of experience in data-driven technologies. He has authored or coauthored several papers in peer-reviewed journals and conferences. His research interests include pattern recognition, machine learning, soft computing and modeling techniques for modern computer systems, and the development of affective computing applications to promote the user's wellbeing.