IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 9, 2022, accepted January 15, 2022, date of publication January 18, 2022, date of current version January 31, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144333

Deep Learning-Based Fault Prediction in Wireless Sensor Network Embedded Cyber-Physical Systems for Industrial Processes

HANG RUAN[1], (Member, IEEE), BOGDAN DORNEANU[2], HARVEY ARELLANO-GARCIA[2], PEI XIAO[3], (Senior Member, IEEE), AND LI ZHANG[4], (Senior Member, IEEE)

[1]School of Mathematics, University of Edinburgh, Edinburgh EH9 3FD, U.K.
[2]LS Prozess- und Anlagentechnik, Brandenburgische Technische Universitaet Cottbus-Senftenberg, 03046 Cottbus, Germany
[3]Institute for Communication Systems, University of Surrey, Guildford GU2 7XH, U.K.
[4]Department of Computer Science, Royal Holloway, University of London, Egham TW20 0EX, U.K.

Corresponding author: Hang Ruan (hr648@outlook.com)

This work was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R001588/1.

ABSTRACT This paper investigates the challenging fault prediction problem in process industries that adopt autonomous and intelligent cyber-physical systems (CPS), which is in line with the emerging developments of industrial internet of things (IIoT) and Industry 4.0. Particularly, we developed an end-to-end deep learning approach based on a large volume of real-time sensory data collected from a chemical plant equipped with wireless sensors. Firstly, a novel recursive architecture with multi-lookback inputs is proposed to perform autoregression on imbalanced time-series data as a preliminary prediction. In this process, a novel learning algorithm named recursive gradient descent (RGD) is developed for the proposed architecture to reduce cumulative prediction uncertainties. Subsequently, a classification model based on temporal convolutions over multiple channels with decay effect is proposed to perform multi-class classification for fault root cause identification and localization. The overall network is named the cumulative uncertainty reduction network (CURNet), for its superior capacity in reducing prediction uncertainties accumulated over multiple prediction steps. Performance evaluations show that CURNet is able to achieve superior performance especially in terms of fault prediction recall and fault type classification accuracy, compared to the existing techniques.

INDEX TERMS Cyber-physical systems, deep learning, fault classification, LSTM, multivariate multi-step prediction, predictive maintenance, TCN, time-series, imbalanced data, uncertainty propagation.

I. INTRODUCTION

The digital transformation is recognised as the most important driver in the era of Industry 4.0. In line with the rapid development of industrial internet-of-things (IIoT), manufacturing industries are equipped with wireless sensors throughout their supply chains, acting as pivots of their cyber-physical systems (CPS), which provide continuous monitoring capabilities. These sensors can be either fixed or mobile, built-in or external, uni-functional or multi-functional. Locally deployed smart sensors which build connections and transmit data to each other from a wireless sensor network (WSN). A WSN is often used to monitor dynamic environments that change rapidly over time, by accessing data using built-in machine learning techniques. However, the challenges of data aggregation, reliability, localization, node clustering, security, fault prediction remain to be addressed in research [1].

In Industry 4.0 and IIoT, a large amount of information and data are generated by WSN deployments in various environments and distributed over scales. This leads to an increasing demand of research on predictive maintenance techniques for the new era of big data and deep learning, which utilize the state-of-the-art approaches to address future IIoT challenges especially on the safety and system security level, such as anomaly and fault detection and early prediction. Such services and deployments may save a huge amount of system and financial resources if any fatal anomaly

The associate editor coordinating the review of this manuscript and approving it for publication was Qi Zhou.

issue or fault can be potentially predicted by the system itself and consequently avoided by taking precautious measures in advance. One type of CPS fault is exclusively incurred by adversarial attacks on the sensor nodes and can be categorized as external faults. Due to the intrinsic vulnerabilities that malicious agents can explore in order to harm the integrity of the CPS, extensive research has been conducted on tackling adversarial attacks under different conditions [2]–[10]. Specifically, the work in [2]–[5] have developed effective algorithms for accurate secure state estimation for distributed multi-agent CPS under adversarial cyber-attacks. [7] developed a two-level detector for deception-based sensor attacks. The work in [9] proposed a new IoT infrastructure against conventional cyber-attacks and its platforms are machine learning compatible. ca10 proposed to exploit physical plant state information to enhance both reliability and security by continuously monitoring the plant state trajectory. The work in [11], [12] study false data-injection attacks in CPS, replying on all historical information to reveal the attacks. The other type of CPS fault is induced only by mechanism failures of internal physical components in the physical layer. In chemical process CPS, continuous measurements of level, pressure, temperature or flow rate from the process units usually reflect a smooth and slow value change in a certain time period when a component failure happens, which must involve multiple time steps to detect the fault in advance depending on the measurement frequency [13]–[16]. In the following, we will focus on the internal CPS fault prediction case, which is to predict chemical unit failures with the help of direct sensory data measurements from those units. Since the operators were not allowed to make physical damages to the units in the plant, we manually injected false data tampered with fluid dynamics equations. However, this data injection process can also be regarded as a kind of cyber-attack even though it is not adversarial but only for modelling the fault characteristics of the physical layer faults. In another word, if a model can successfully predict from injected false data, then it also proves to be an effective solution against the false data injection type of cyber-attacks or attacks that result in an underlying long-time-range effect on the data.

While efficient big data handling tools have been developed to cope with data generated from future IIoT applications [17]–[19], machine learning has been well known for learning underlying patterns from data and thereafter making insightful predictions for difficult tasks in complex scenarios. The conventional machine learning classification algorithms including support vector machines (SVM), K-nearest neighbors (KNN), Gaussian mixture model (GMM) have been employed to provide satisfactory performances [20]–[25] when the data size is small to moderate, with additional expert domain knowledge or reasonable assumptions on the distribution of fault data. A better solution is to employ highly intelligent deep learning models that are capable of performing data analysis tasks continuously and reliably.

In multi-step predictions, uncertainties accumulate over the steps and lead to significant prediction errors. In [16], a systematic approach that combines operable adaptive sparse identification of systems (OASIS) and dynamic risk assessment is proposed to realize multi-step prediction. In the illustration experiment, their method is able to predict a maximum of eight steps ahead for reactor over-pressure caused by the change of temperature. However, their method strongly relies on control system designs and effective risk assessment analysis and there is no indication whether their dataset is imbalanced or not regarding the fault data ratio. Studies which use deep learning based approaches can be found regarding time-series multi-step prediction while dealing with cumulative errors: as an example, in [26], the authors attempt to test multi-step prediction performance using reccurent LSTM network on different data patterns. However, it is clearly stated that their prediction accuracy decreases drastically as the time steps increases. In [27], the authors are able to perform up to eighteen steps ahead for multi-step prediction with minimum error rates using their datasets. Their method requires data distribution with strong and clear seasonality variations, where industrial process data usually do not have. The studies in [28] and [29] compare different time-series prediction strategies for a single LSTM unit prediction whereas [30] proposes a stacked LSTM network using multiple units to maintain higher prediction capacity. However, the scope of those studies are simply on reducing autoregression errors rather than making precise fault prediction. The work in [31] proposes an deep architecture by stacking LSTM-autoencoders with internal cooperation for anomaly prediction tasks in industrial processes. But this proposed architecture has unknown complexity and does not focus on the imbalanced data distributions.

In predictive maintenance, the majority of data usually indicates a normal state of operation, whereas only a very small fraction of the data indicates underlying or potential faults. This class out-weighting imbalance limits the success of data-driven based machine learning methods in predicting faults, thus presenting a significant hindrance in the progress of smart manufacturing and is a major challenge for ML-based predictive analytics which rely on data for learning. To work around this issue, efforts have been made on data preprocessing (using techniques like windowing, normalization [32]) and selecting imbalanced data (using a separate deep reinforcement learning method [33]), which adds extra burden and opacity to the processes. In [34] and [35], machine learning fault prediction models for imbalanced industrial processes have been developed, which however still lack capacity for long time range prediction. The other deep learning models [36]–[39] have shown satisfactory anomaly prediction results including prediction time ranges, but their applicability on industrial process data remain unknown.

In order to reduce the efforts of domain-specific system designs, a real-time end-to-end deep learning framework must be developed to capture the long-range time dependencies of features to predict future faults on realistically imbalanced fault ratio datasets obtained from industrial processes. In this work, an interdisciplinary modular CPS system is

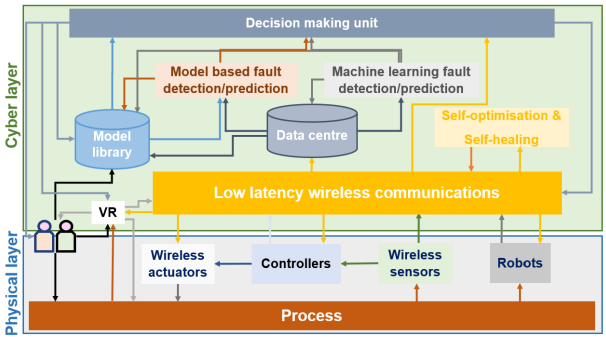


FIGURE 1. Proposed high-level CPS model for industrial process monitoring.

introduced to emulate intelligent and autonomous industrial process monitoring with large capabilities. The focus of the following sections is on the development of deep learning strategies for fault prediction applications. The main contributions of this paper include the following:

- A recursive deep architecture with multi-lookback inputs is proposed. To the best of our knowledge, this specific deep architecture has not been previously investigated in industrial process applications.
- We highlight that, as a main contribution, a novel learning algorithm, named recursive gradient descent (RGD), which is both theoretically analyzed and empirically tested to be highly effective at reducing cumulative prediction uncertainties for the proposed architecture, providing fast learning and convergence rates in the system training.
- Subsequently, a classification model based on temporal convolutions over multiple channels with decay effect is proposed to perform accurate multi-class classification for fault root cause identification and localization, where a flexible re-sampling process is also introduced.
- Finally, by integrating all the components, a novel data-driven and data-based deep network (recurrence-first, convolution-second) named the cumulative uncertainty reduction network (CURNet) is developed, which gives superior performance in both fault prediction and classification on imbalanced data.

The remainder of this paper is organized as follows: Section II introduces the case study of the high-level industrial CPS model. Section III introduces the modified multi-lookback LSTM unit. Section IV introduces the proposed CURNet. Section V illustrates the performance metrics based on numerical simulation results. Section VI discusses the main conclusions and remarks.

II. CASE STUDY

A. THE CPS MODEL DESIGN

Specifically in the CPS system design, there are five modules that constitute this envisioned infrastructure as shown in Fig. 1: the holistic system models and cooperative control, massive connectivity and resilient network communication, machine learning-based fault detection and prediction, an intelligent adaptive decision making framework, and a virtual reality system for visualization. This modular infrastructure represents the complexity of industrial processes with a large number of interconnected units operating in extreme environments, which includes hazardous gas diffusion and material leakage, over-pressured tanks, power outage, relating to a paramount health and safety practice. The vertical integration of the modules inside the plant to implement a flexible and reconfigurable production system represents a blueprint of enhanced safety integrity in industrial settings with CPS embedded at its core which is centred on plant control and management of a real plant through parallel operation and interaction of actual computing and operational functions defined by software. It extends traditional simulations and attempts to realize the control and management of actual and virtual plant through IIoT in a timely manner.

In the physical layer, the wireless sensors directly and continuously take process measurements and share firsthand data with the controller. Meanwhile, an artificial mobile robot equipped with sensing hardware can also be employed for periodic maintenance, surveillance, monitoring and to act as sensor relays in deep fading areas where they are handled as mobile multi-functional wireless sensors in the WSN. In the cyber layer, a low-latency wireless communication network is utilised to minimize the data transmission delay and overhead. The measurement data are transmitted to the data centre where the new data are stored along with the historical data. Then, two independent sets of models take continuous data inputs from the data centre, in order to perform both model based and machine learning based fault detection and prediction functionalities, by interactively accessing historical data required for making predictions from the data centre. The decision-making unit will update the model library based on the outputs of those two blocks and instruct the process on which actions could/should be taken. These decisions are transmitted via the wireless communication network. In addition, a virtual reality (VR) interface is utilized to facilitate human operators to intervene through the cyber layer to tackle necessary emergency situations as well as for remote surveillance and control purposes of the physical layer.

For the wireless sensors, it is envisioned that a new design paradigm is needed to support large numbers of heterogeneous sensing devices with diverse requirements and unique traffic characteristics. Comparing to the sensors in traditional IoT networks, those deployed in extreme environments need to operate in harsh (sometimes hazardous) conditions and are prone to wear and tear, and cannot be easily replaced, posing major challenges in designing resilient networks for robust communications [40]. Our work assumes a centralised control mechanism where sensors are connected to a fusion node via wireless links. The wireless links can also be used to send commands to actuators within the plant. The network consists

of a heterogeneous set of periodic and event-triggered sensors with mixed requirements, characteristics and traffic models.

For the machine learning component, the data center works as a database that keeps a record of stable and continuous data flow via receiving data from wireless sensors and delivering it to the model and machine learning based predictors, which are embedded for existing fault type look-ups and new fault predictions, respectively. Both predictors are software (Python) defined and take input data streams into computation as soon as they arrive to minimize the processing latency and maintain continuous real-time predictions.

B. CPS FAULT MODEL AND PREDICTION CHALLENGES

The techniques employed for process monitoring and fault detection have been primarily dominant by data-driven methods [41], [42]. These techniques rely on huge sets of historical process data and often require little insight into the system. Furthermore, they tend to be less affected by the presence of noise and perform well within the range of the collected data. In process industry, virtual sensing technology is often used to construct inference models. Multiple linear and partial least square regression are the most popular approaches; however, they are not appropriate for modelling observed defects. To overcome the limitations of these methods, Poisson regression, a discrete probability distribution that expresses the probability values of non-negative integers, is widely applied [43]. Therefore, we opt for modelling the fault effect by injecting a number of Poisson data distribution at spread time intervals, to model rapid process responses as anomalous faults. Specifically, for the input-output sequence $\{(X_m, y_m)\}$ ($X_m \in \mathcal{R}^N$ where N is the feature variable dimension), the Poisson distribution $p(y_m|X_m) = \frac{e^{-\lambda}\lambda^{y_m}}{y_m!}$ models the probability of a certain number of faults to happen within a fixed time interval, which is defined by the rate parameter λ. In our case, we aim to control the number of false data at a very small ratio to model imbalance, and leave them in a number of separate time intervals of centralized data points. These time intervals have the same length where the data points in each of them are modelled with Poisson distribution to generate faults whose values are randomly scaled by a factor of *alpha*. The faults are also supposed to be detectable if they have taken place for a while and caused consequential harms in real-time, by three additional system state variables ''connection'', ''Alarm'' and ''E-Stop'', all of which are in Boolean values and used to indicate the operating states of the CPS. The faults are only injected to the temperature, level and one flow variables at different locations. These injected faults are expected to correlatively emulate the long-time-range dispersion behavioral characteristic of the system relating to unexpected system state changes.

Fault prediction in predictive maintenance plays a key role in the high-level CPS of industrial processes and usually values systems that are rather fault-sensitive than fault-tolerant. This is because the principle of fault-tolerant predictions is to focus on major fatal faults, but to neglect non-fatal faults or glitches unless they escalate to a more serious level, which is out of financial cost and maintenance complexity consideration to deal with small faults every time as they arise. Oppositely, the principle of fault-sensitive predictions requires the system to capture and eliminate underlying faults as many as possible, regardless of their levels of harm. Fault-sensitive systems are usually highly vulnerable and require stringent fault prediction methods to avoid any inevitable and disastrous consequences to the system. Another challenge is that datasets recorded from industrial processes are usually highly imbalanced, with a very small fraction of the data indicating faults, since systems operate in normal states for the majority of time. This results in increased difficulties for machine learning models to learn different fault types efficiently prior to making predictions. On the other hand, designing any end-to-end DNN in seek of a simple and unified solution for CPS is practically infeasible, due to the heterogeneous nature of CPS and non-stationary environmental effects. Multi-step predictions are essential for most process industries to earn sufficient time to take actions to prevent faults from happening. However, reliability and performance of the existing multi-step prediction methods are significantly degraded due to the cumulative prediction uncertainties carried over from multiple prediction steps. For the above reasons, we develop a novel end-to-end deep learning data-based data-driven approach to automatically and effectively learn from both historical and new data and thereby predict the underlying system faults with high sensitivity, which present a long-time-range dispersion characteristic.

III. MODIFIED LSTM UNIT WITH MULTI-LOOKBACK INPUTS

In the section, we introduce a modified LSTM unit which can take multiple lookbacks as its input at once. We use mathematical symbol notations to represent quantities in the proposed architecture. Specifically, vectors and matrices are denoted by lowercase and uppercase bold letters, respectively. The time index and data sample index are represented by t and m, respectively.

A single LSTM unit designed for multi-lookback inputs (a windowed input frame covering more than one time step) without peephole connections is considered as shown in Fig. 2.

Assuming K is the number of lookbacks and N is the number of raw features denoting the sensor measurement types (i.e., level, pressure, temperature or flow rate) at different placement locations. Then we have

$$X_m(t) = [x_m^T(t),\ x_m^T(t-1), \cdots, x_m^T(t-K+1)]^T, \quad (1)$$

where $X_m(t) \in \mathbb{R}^{K\times N}$ and $x_m(t-k+1) \in \mathbb{R}^{1\times N}$ $(k = 1, \cdots, K)$ is the input. $y_m(t) \in \mathbb{R}^{1\times h}$ is the output. i, o and f denote the input gate, output gate and forget gate subscripts, respectively. $c(t) \in \mathbb{R}^h$ is the cell state vector. The unit parameters include the weight matrices $W_i, W_o, W_f, W_c \in \mathbb{R}^{h\times N}$ and $U_i, U_o, U_f, U_c \in \mathbb{R}^{h\times h}$ for the input and recurrent vectors, respectively; the bias vectors $b_i, b_o, b_f, b_c \in \mathbb{R}^h$, the

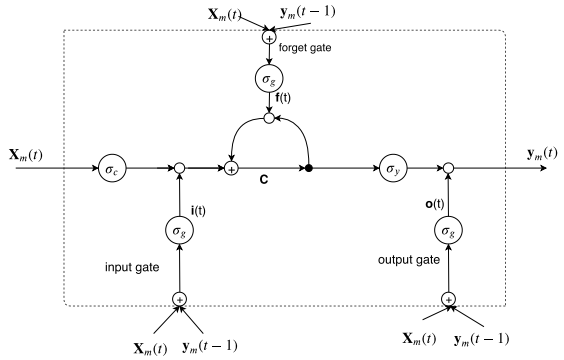


FIGURE 2. A single LSTM unit with multi-lookback inputs.

activation functions $\sigma_g, \sigma_y, \sigma_c \in \mathbb{R}^h$ for the gates, output and input are usually set as $sgm(\cdot)$, $\tanh(\cdot)$ and $\tanh(\cdot)$, respectively. The following equations describe their relationships:

$$i(t) = \sigma_g\Big(\phi_i\big(W_i X_m^T(t), K\big) + U_i y_m^T(t-1) + b_i\Big), \quad (2)$$

$$o(t) = \sigma_g\Big(\phi_o\big(W_o X_m^T(t), K\big) + U_o y_m^T(t-1) + b_o\Big), \quad (3)$$

$$f(t) = \sigma_g\Big(\phi_f\big(W_f X_m^T(t), K\big) + U_f y_m^T(t-1) + b_f\Big), \quad (4)$$

$$c(t) = f(t) \circ c(t-1) + i(t) \circ \sigma_c\Big(\phi_c\big(W_c X_m^T(t), K\big) + U_c y_m^T(t-1) + b_c\Big), \quad (5)$$

$$y_m(t) = [o(t) \circ \sigma_y\big(c(t)\big)]^T, \quad (6)$$

where $\circ$ denotes the Hadamard product, $(\cdot)^T$ denotes matrix transpose, $\phi_i(\cdot)$, $\phi_o(\cdot)$, $\phi_f(\cdot)$ and $\phi_c(\cdot)$ are the lookback aggregation functions that compute a vector of dimension $\mathbb{R}^h$. For instance, when $\phi_*(\cdot, K)$ $(* \in \{i, o, f, c\})$ is chosen as linear aggregation of K:

$$\phi^*(W^* X_m^T(t), K) = \sum_{k=1}^{K} W^* X_m^T[:, k](t), \quad (7)$$

where $X_m^T[:, k]$ refers to the kth column of matrix X_m^T.

The above described LSTM unit will be used as a component to built the proposed CURNet in the following section.

IV. THE PROPOSED CURNET

This section introduces the proposed CURNet architecture illustrated in Fig. 3, where the time index t is omitted for simplicity. The CURNet is designed in a novel two-phase architecture which puts multi-level recurrences at phase one and convolutions at phase two, distinguishable from all existing architectures [26]–[39], which (if any) put convolutions at first and recurrences at second. The multi-level recurrences include state self-recurrence of each individual LSTM unit and an overall feedback from output to input of a fully connected LSTM network. The decayed multi-channel convolutions subsequently take the LSTM network outputs as input and apply a time-based decay effect on them before extracting and categorizing faults from convolutions. The multi-level recurrences recursively eliminate the error accumulations in small steps with the help of the RGD learning algorithm which significantly reduce the obscurity of hidden faults and expose them from the majority of normal data, and also facilitate the fault classification task in the convolution phase. The CURNet showcases an effective end-to-end deep learning architecture without any preliminary feature extractions and can be directly built into the machine learning module in the aforementioned CPS system model.

A. A RECURSIVE DEEP ARCHITECTURE WITH MULTI-LOOKBACK INPUTS

The system input $X \in \mathbb{R}^{m \times K \times N}$ is considered as an online 3D data sample sequence given by

$$X = [[X_1], [X_2], \cdots, [X_m]]. \quad (8)$$

Each sample X_m is composed of K consecutive observations in a time frame of the latest K (including the current) time steps as defined in (1). Without losing generality, only one data sample X_m is assumed to be input to the system at each time step t throughout the online process. The output 3D sequence $Y \in \mathbb{R}^{m \times L \times N}$ is represented by

$$\hat{Y} = [[\hat{Y}_1], [\hat{Y}_2], \cdots, [\hat{Y}_m]]. \quad (9)$$

Each output sample $\hat{Y}_m \in \mathbb{R}^{L \times N}$ composes of L instantaneous prediction vectors as

$$\hat{Y}_m(t) = [\hat{y}_m^T(t+1), \cdots, \hat{y}_m^T(t+L)]^T, \quad (10)$$

where L is the number of prediction time steps, $\hat{y}_m(t+l) \in \mathbb{R}^{1 \times N}$ $(l = 1, \cdots, L)$.

Instead of constructing a layout of RNN based connections like in the previous work [44], we build full connections between every two consecutive hidden layers of LSTM units with multi-lookback inputs, whilst also enable feedback at different phases, which can be seen in Fig. 3. It is remarked that there is no inter-layer connection between the LSTM units of the same layer. In order to allow multi-step prediction, the output feedback $\hat{y}_m^T$ recursively allows a maximum of L recursions to take place exactly once for any given input sample X_m. For the sake of simplicity, the time index t is dropped in the equations as well. The system input after l $(l = 1, \cdots, L-1)$ recursions is updated as

$$X_m(l+1) = [X_m^T(l)[:, 1:], \hat{y}_m^T(l+1)]^T, \quad (11)$$

where $X_m^T(l)[:, 1:]$ refers to the second to the last columns (excluding the first column) of $X_m^T(l)$, which is the system input in the lth recursion, $\hat{y}_m^T(l+1)$ is the system output in the lth recursion. Once the recursions are completed, the output is a stack of $\hat{y}_m(l)$, $l = 1, \cdots, L-1$ given by

$$\hat{Y}_m = [\hat{y}_m^T(1), \cdots, \hat{y}_m^T(L)]^T, \quad (12)$$

Specifically, for a given input sample $X_m \in \mathbb{R}^{K \times N}$, an L-step $(1 \leq L < K)$ ahead prediction is given as in (10), with an objective loss function $\mathcal{J}_1(\hat{Y}_m(t), Y_m(t))$, where $Y_m(t)$ is the ground truth output given by

$$Y_m(t) = [y_m^T(t+1), \cdots, y_m^T(t+L)]^T, \quad (13)$$

$$y_m(t+l) = x_m(t+l+1), \; l = 0, \cdots, L-1. \quad (14)$$

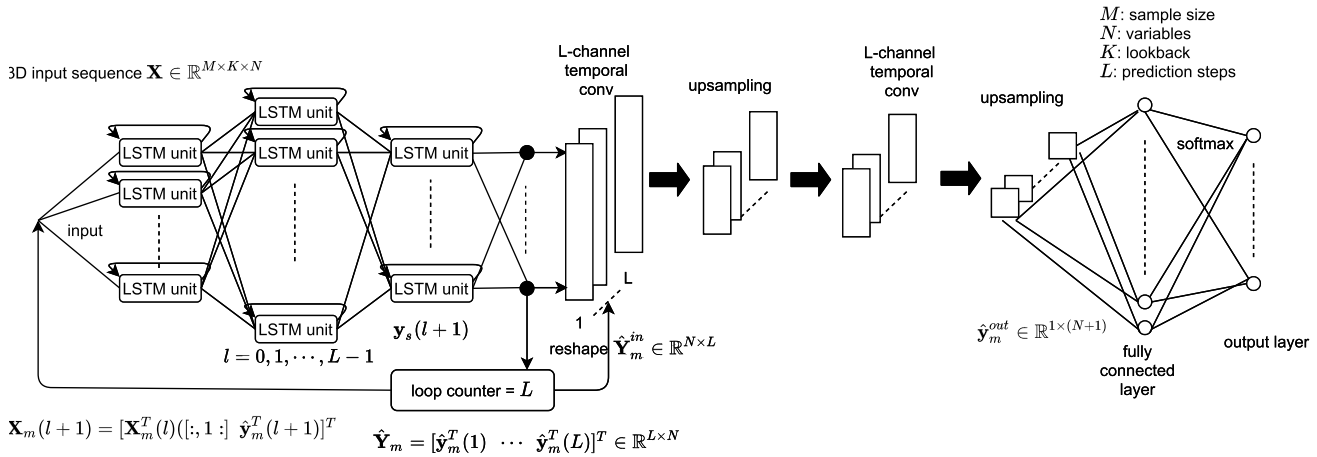


FIGURE 3. The proposed CURNet architecture.

When SGD is applied as the learning algorithm, the instantaneous loss function becomes

$$\mathcal{J}_1\big(\hat{Y}_m(t), Y_m(t)\big) = ||\hat{Y}_m(t) - Y_m(t)||_F, \tag{15}$$

where $||\cdot||_F$ denotes the Frobenius norm of a matrix.

B. A CLASSIFICATION MODEL WITH TEMPORAL CONVOLUTIONS

As shown on the right-hand side of Fig. 3, a subsequently concatenated temporal classifier is proposed for fault classification. The L channels take different column components with consecutive time delays from the previous output as input, in order to facilitate learning the temporal correlations from the preliminary prediction results. The input to the first convolution layer is reshaped into $\hat{Y}_m^T \in \mathbb{R}^{N\times L}$ and the input to the lth channel becomes $\hat{y}_m^T(l)$, $l = 1, \cdots, L$. As an additional step, we introduce the idea of using a channel decay factor which, inversely puts less weights on the predictions farther to the current time step. Denoting the predicting factor as $p \in (0, 1)$, the actual input to the lth channel $\hat{y}_m^{in}(l)$ with channel decay effect is written as

$$\hat{y}_m^{in}(l) = p^{l-1}\hat{y}_m^T(l), \quad l = 1, \cdots, L. \tag{16}$$

We remark that p is often chosen to be close to 1 to prevent information losses. The channel decay modeled based on a predicting factor as in (16) aims to mitigate the weights of the prediction uncertainties that gradually accumulate over time. The further the prediction step is, the less weight and more channel decay are imposed to the corresponding channel.

Assume the filter for the lth channel is $d_l \in \mathbb{R}^{F_w\times 1}$ ($l = 1, \cdots, L$), where F_w is the length of the filters and the stride is chosen as 1, a padding of $\frac{F_w-1}{2}$ on both sides of $\hat{y}_m^T(l)$ is applied for all channels to ensure the output of the convolution remains in the same length as the input $\hat{y}_m^T(l)$. Then the convolution output for the lth channel of the first layer is given as

$$d_l[n]\hat{y}_m^{in}(l)[n] = \sum_{i=0}^{N-1} d_l[i]\hat{y}_m^{in}(l)[n-i], \quad l = 1, \cdots, L, \tag{17}$$

which is followed by an activation function, for instance, the ReLu function $ReLu(y) = max(0, y)$ and upsampling. Further, another temporal convolution block is implemented for each channel, after which a fully connected layer is utilized to learn the long range dependencies of multiple time steps. Finally, in order to address such a multi-class classification problem, a softmax function is implemented as the last layer of the model before the output layer to normalize the generated probability scores from the previous to the range of 0 to 1 for each assigned individual class. The class type with the highest probability presenting the highest likelihood will be categorized and predicted as the model output: at any given time step, if the model finds no fault, then the no-fault class will present the highest probability score close to 1 with the others close to 0; if the model detects a potential system fault, then a specific fault class will output a higher probability close to 1 with the no-fault class close to 0. The real-time online temporal classification is based on the preliminary prediction $\hat{Y}_m(t)$ from phase one. Provided the fact that faults can happen in any process unit at different placement locations, a $(N+1)$-class classification problem is considered. This allows to perform root cause identification which also relates to the exact location where the fault is most likely going to take place. Let us denote the system output ground truth as $y_m^{out} \in \mathbb{R}^{1\times N}$ with element values belong to the Boolean set $\{1, 0\}$, the goal is to minimize the categorical cross-entropy loss function defined as

$$\mathcal{J}_2 = -\sum_{n=1}^{N+1} y_m^{out}[n] \cdot \log\big(softmax(c_n)\big), \tag{18}$$

where $y_m^{out}[n]$ represents the nth element of y_m^{out}, c_n is the probability score of class n, $softmax(c_n) = \frac{e^{c_n}}{\sum_{i=1}^{N+1} e^{c_i}}$ is the softmax function.

The dataset will be split to 60%, 15% and 25% for training, validation and testing, respectively. Considering the end-to-end training difficulties of a heterogeneous deep network like CURNet, it is more feasible to split training into two separate phases. In the first phase, the recursive deep architecture is

trained for multi-step autoregression. That is, we will train 60% of the input X_m and output Y_m pairs for the first phase and validate the network using the rest 15% of the total training data. Once training is completed, the other 25% data will be used for testing. In the second phase, we will directly use the actual output of phase one (i.e., $\hat{Y}_m$) as input, and with the corresponding output vector as y_m^{out}, where the training, validation and test ratios remain the same. For the sake of simplicity, the *train*, *val* and *test* subscripts are omitted as they are standard machine learning procedures.

C. RECURSIVE GRADIENT DESCENT

In this subsection, a novel gradient descent based learning algorithm is developed for the proposed CURNet architecture, namely, the recursive gradient descent (RGD) algorithm, which updates the weights for both inner recursions and outer iterations, is highly effective at reducing cumulative prediction uncertainties for the proposed architecture and converging faster comparing to its counterpart SGD. In the following, we will derive the RGD algorithm as the learning algorithm for phase one of the CURNet. First of all, we model the loss function for the recursive architecture based on the sum of weighted least squares functions associated with the previously introduced predicting factor p, which is defined as

$$\mathcal{J}_1^{RGD}(t,r) = \sum_{l=1}^{r} p^{l-1} ||\hat{e}(t+l)||_2^2, \tag{19}$$

where $r = 1, \cdots, L$, $r \leq L$, $||\cdot||$ denotes the norm operator and

$$\mathcal{J}_1^{RGD}(t,r+1) = \mathcal{J}_1^{RGD}(t,r) + p^r ||\hat{e}(t+r+1)||_2^2. \tag{20}$$

$$\hat{e}(t+l) = \hat{y}_m(t+l) - y_m(t+l-1), \quad l = 1, \cdots, r, \tag{21}$$

is the prediction uncertainties at time step t and lth internal recursion. By submitting (14) in (21), we have

$$\hat{e}(t+l) = \hat{y}_m(t+l) - x_m(t+l), \quad l = 1, \cdots, r. \tag{22}$$

For simplicity, let us denote an LSTM unit function as

$$\xi(t) = \hat{y}_m(t+1) = \xi\big(X_m(t+1), \hat{y}_m(t), W^*(t), U^*(t)\big), \tag{23}$$

where $* \in \{i, o, f, c\}$. From Fig. 3, an LSTM unit in any of the hidden layers of the recursive deep LSTM takes outputs from all units from the previous layer and the output of itself as inputs. If the inter-layer connections are unweighted, it yields

$$x_m(t) = \frac{1}{a} \sum_{j=1}^{a} \hat{y}_m^j(t-1), \tag{24}$$

where a is the number of LSTM units in the previous layer and $\hat{y}_m^j(t-1)$ is the output of the jth unit from that layer. $x_m(t)$ and $X_m(t)$ are bounded by equation (1). To this point, we take the partial derivatives of $\mathcal{J}_1^{RGD}(t,r)$ in (19) with respect to the LSTM weight parameters $W^*(t)$ and $U^*(t)$:

$$\begin{aligned}
&\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial W^*(t)} \\
&= \frac{\partial \sum_{l=1}^{r} p^{l-1} ||\hat{e}(t+l)||_2^2}{\partial W^*(t)} \\
&= \frac{\sum_{l=1}^{r} p^{l-1} \partial ||\hat{e}(t+l)||_2^2}{\partial W^*(t)} \\
&= \frac{\sum_{l=1}^{r} p^{l-1} \partial ||\hat{y}_m(t+l) - x_m(t+l)||_2^2}{\partial W^*(t)} \\
&= 2 \sum_{l=1}^{r} p^{l-1} ||\hat{y}_m(t+l) - x_m(t+l)||_2 \frac{\partial \hat{y}_m(t+l)}{\partial W^*(t)} \\
&= 2 \sum_{l=1}^{r} p^{l-1} ||\hat{y}_m(t+l) - x_m(t+l)||_2 \frac{\partial \xi(t+l-1)}{\partial W^*(t)}.
\end{aligned} \tag{25}$$

Similarly for $U^*(t)$, we obtain

$$\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial U^*(t)} = 2 \sum_{l=1}^{r} p^{l-1} ||\hat{y}_m(t+l) - x_m(t+l)||_2 \times \frac{\partial \xi(t+l-1)}{\partial U^*(t)}. \tag{26}$$

Since both the terms $\frac{\partial \xi(t+l-1)}{\partial W^*(t)}$ and $\frac{\partial \xi(t+l-1)}{\partial U^*(t)}$ can be computed by expanding the equations (2) to (6) depending on the structure of the LSTM unit and are independent from the RGD learning algorithm itself, the related equation breakdowns are omitted for simplicity. As the next step, we aim to introduce recursive weight updates within each time step based on the loss function as defined in (19), so that the weight gradients are also internally updated in order to achieve more accurate estimations along with the multi-step predictions. Therefore, at a given time step t, we define

$$W^*(t,r) = W^*(t,r-1) - \mu_{W^*} \frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial W^*(t,r-1)}, \tag{27}$$

$$U^*(t,r) = U^*(t,r-1) - \mu_{U^*} \frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial U^*(t,r-1)}, \tag{28}$$

where $W^*(t,r-1)$ and $U^*(t,r-1)$ represent the LSTM unit weights in the tth time step and rth prediction step, μ_{W^*} and μ_{U^*} are the learning rates (step sizes) for $W^*(t,r-1)$ and $U^*(t,r-1)$, respectively. $\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial W^*(t,r)}$ and $\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial U^*(t,r)}$ are computed from (25) and (26) directly after replacing $W^*(t)$ and $U^*(t)$ with $W^*(t,r-1)$ and $U^*(t,r-1)$, respectively. As soon as a total of L internal recursions are finished, the weights for the next time step are simply updated by the last recursion in the current time step:

$$W^*(t+1, r=0) = W^*(t, r=L), \tag{29}$$

$$U^*(t+1, r=0) = U^*(t, r=L). \tag{30}$$

The RGD algorithm is summarized in Table 1.

One of the main differences of the RGD algorithm compared to conventional SGD algorithm is the loss function

TABLE 1. The RGD learning algorithm.

Initialization:
$W^*(t=0,r=0)$ ($W_i(t=0,r=0), W_o(t=0,r=0),$
$W_f(t=0,r=0), W_c(t+0,r=0)$);
and $U^*(t=0,r=0)$ ($U_i(t=0,r=0), U_o(t=0,r=0),$
$U_f(t=0,r=0), U_c(t=0,r=0)$);
$p=0.95$; $\mu_{W^*}=0.01$; $\mu_{U^*}=0.01$.
for $t=0,1,\cdots$:
 for $r=1,2,\cdots,L$:
 $\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial W^*(t)} = 2\sum_{l=1}^{r} p^{l-1}||\hat{y}_m(t+l)-x_m(t+l)||_2 \frac{\partial \xi(t+l-1)}{\partial W^*(t)}$
 $W^*(t,r) = W^*(t,r-1) - \mu_{W^*}\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial W^*(t,r-1)}$
 $\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial U^*(t)} = 2\sum_{l=1}^{r} p^{l-1}||\hat{y}_m(t+l)-x_m(t+l)||_2 \frac{\partial \xi(t+l-1)}{\partial U^*(t)}$
 $U^*(t,r) = U^*(t,r-1) - \mu_{U^*}\frac{\partial \mathcal{J}_1^{RGD}(t,r)}{\partial U^*(t,r-1)}$
 End for;
 $W^*(t+1,r=0) = W^*(t,r=L)$
 $U^*(t+1,r=0) = U^*(t,r=L)$
End for;

as defined in (19). The loss function takes a recursive form from the error term $\hat{e}$ and sums over a total of L sub-time steps at time step t before moving to step $t+1$. This loss function mimics the accumulated errors piled up through the feedback and loop counter (as shown in Fig. 3). Then during training, the system weights are adjusted by minimizing this loss function at every r-step ($0 \le r \le L$) for every t-step. In another word, RGD allows the system weights updated from minimizing the accumulated error at smaller r-step than t-step, as presented in (27) and (28), which demonstrates the strength of RGD at learning accumulated prediction errors.

To study the convergence of RGD, the following assumptions are required as $\forall\ t \ge 0, t \to \infty$, ($a$): the loss function $\mathcal{J}_1^{RGD}(t,r)$ has a lower bound (i.e., $\mathcal{J}_{1,min}^{RGD}$); ($b$): let the loss function gradient be $||\nabla_{\mathcal{J}_1^{RGD}}(t,r)||$, then the gradient variance is constrained, i.e., $\mathbb{E}[||\nabla_{\mathcal{J}_1^{RGD}}(t,r)||^2] \le \sigma_\nabla^2$. Note that, similarly for the conventional gradient descent algorithm, norm constraint requirements are often necessary to guarantee convergence (e.g., [45] utilizes gradient descent to build a new online learning policy for real-time output feedback dynamic controls for fuzzy systems, which models a multitude of norm control effect and further promotes the sufficiency of its cost function convergence; [46] also studies the conventional SGD convergence in the multivariate case by assuming the $\mathcal{L}$-Lipschitz continuity). Let μ_{W^*} and μ_{U^*} equal the same fixed step size μ^*, for any recursion, we have

$$\begin{aligned}&\mathcal{J}_1^{RGD}(t,r)\\ &\le \mathcal{J}_1^{RGD}(t,r-1)\\ &\quad -\mu^*||\nabla_{\mathcal{J}_1^{RGD}}(t,r-1)|| + \mu^*||\nabla_{\mathcal{J}_1^{RGD}}(t,r-1)||^2, \quad (31)\end{aligned}$$

whose both sides can be summed up over both r and t and yields the following after rearranging

$$\begin{aligned}&\sum_{k=0}^{t-1}\sum_{r=1}^{L}\mathcal{J}_1^{RGD}(k,r) - \sum_{k=0}^{t-1}\sum_{r=1}^{L}\mathcal{J}_1^{RGD}(k,r-1)\\ &\le \mu^*\sum_{k=0}^{t-1}\sum_{r=1}^{L}\left(||\nabla_{\mathcal{J}_1^{RGD}}(t,r-1)||^2 - ||\nabla_{\mathcal{J}_1^{RGD}}(k,r-1)||\right). \quad (32)\end{aligned}$$

Taking expectations on both sides and cancelling out terms on the left using $\mathcal{J}_1^{RGD}(t+1,r=0) = \mathcal{J}_1^{RGD}(t,r=L)$, we obtain

$$\begin{aligned}&\mu^* Lt\mathbb{E}[||\nabla_{\mathcal{J}_1^{RGD}}(t,r-1)||]\\ &\le \mathcal{J}_1^{RGD}(t=0,r=0) - \mathbb{E}[\mathcal{J}_1^{RGD}(t,r=0)]\\ &\quad +\mu^* Lt\mathbb{E}[||\nabla_{\mathcal{J}_1^{RGD}}(t,r-1)||^2]. \quad (33)\end{aligned}$$

Noting that $\mathbb{E}[\mathcal{J}_1^{RGD}(t,r=0)] \ge \mathcal{J}_{1,min}^{RGD}$ and $\mathbb{E}[||\nabla_{\mathcal{J}_1^{RGD}}(t,r-1)||^2]$ based on assumption (a) and (b) respectively, we obtain

$$\begin{aligned}&\mathbb{E}[||\nabla_{\mathcal{J}_1^{RGD}}(t,r-1)||]\\ &\le \frac{\mathcal{J}_1^{RGD}(t=0,r=0) - \mathcal{J}_{1,min}^{RGD}}{\mu^* Lt} + \sigma_\nabla^2. \quad (34)\end{aligned}$$

This means the RGD algorithm is guaranteed to converge when $t \to \infty$. In addition, the L in-recursion updates along r greatly reduces the iterations required for t to converge. As a result, the RGD algorithm requires less epochs for training and converges faster than the conventional SGD algorithm for the proposed model, which is also illustrated in the Simulation section.

V. PERFORMANCE EVALUATIONS

In this section, the proposed CURNet architecture along with the RGD algorithm is evaluated using a large volume of real-time sensory data collected from the chemical plant, which is equipped with the necessary hardware and software components illustrated in the CPS in Fig. 1, and integrated with essential functions to perform data measurements and storage. The data were collected by site operators between years 2017 and 2018, and has over 135k (i.e., $M > 135k$) samples where each sample is featured by variable measurements of 40 internally deployed fixed sensors, i.e., $N = 40$ in this case. The measurement types include temperature, flow rate, pressure and level whereas the data types include both floating point and Boolean values, with the presence of a small number of nulls. The data samples were collected at a frequency of $F_s = 0.2$ Hz (i.e., one sample every $T_s = 5$ seconds). The raw data are preprocessed to exhibit its best coherency before being transferred to time-series format. In order to take a data-driven approach without relying on any cloud server, a local computer equipped with an Intel i7-8700 CPU (6-Core/12-Thread, 12MB Cache, up to 4.6GHz with Intel Turbo Boost Technology) and dual NVidia GeForce GTX 1080Ti GPUs are used. The simulations are carried out in Python 3.6 with Keras and Tensorflow frameworks.

We choose the re-sampling frequency as twice as the original measurement frequency which gives 0.1Hz for generating the input sequence X, where $\delta_t = 2$ is the data sample gap between two consecutive inputs. Note that if a lower

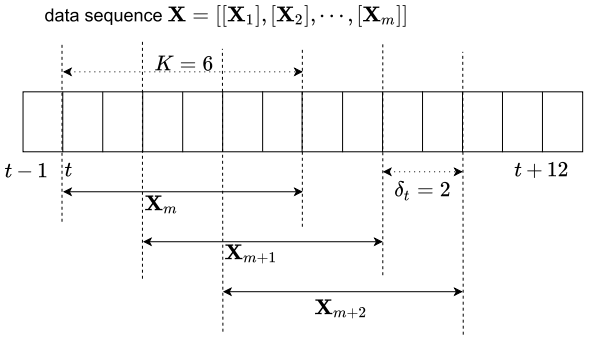


FIGURE 4. Re-sampling process to generate input data sequence.

re-sampling frequency is used, the total number of new data samples will be further reduced. Fig. 4 demonstrates the re-sampling technique, in which the re-sampling frequency $\delta_t = 2$ and number of lookbacks $K = 6$. Additionally, the fault data are controlled at 1% of all data samples. Four different fault injections (two temperature sensors at different locations as well as one level sensor and one flow sensor) following the Poisson distribution with $\lambda = 1$ to indicate one fault per single time step. The scaling effect is applied on false data by $\alpha \in (0, 2)$.

The compared techniques include the state-of-the-art time-series/sequence based fault prediction deep networks as well as the conventional deep LSTM network, in terms of fault prediction performance in our very application. Specifically, we compare DeepAnT [36], FuseAD [37], TwitterAD [38], ConvLSTM in [39] and TCN [47] with the proposed CURNet. The architectures of DeepAnT, FuseAD, TwitterAD, ConvLSTM and TCN are the same as that proposed in their original studies, with a reasonable number of trials in hyper-parameter tuning. Note that, there are a very limited number of existing multivariate multi-step fault prediction models in the literature and most of them suffer from cumulative uncertainties. However, the proposed CURNet works well because the RGD algorithm learns to improve the preliminary prediction result from within the recursions, whereas the temporal classifier also learns long-range time dependencies in the temporal patterns on top of the preliminarily accurate prediction results.

Firstly, we evaluate the autoregression performance of the compared techniques. We use 75% of the total samples as training (including validation) data while leaving the remaining 25% for testing in the online autoregression. The number of lookbacks and prediction steps are set at $K = 8$ and $L = 6$ (which will give $L \cdot \delta_t \cdot T_s = 6 \times 2 \times 5 = 60$ seconds for the system to take emergent actions to prevent any fault), respectively. For the proposed recursive deep LSTM network, we assign $p = 0.95$, $\mu_{W^*} = \mu_{U^*} = 0.01$ and a total number of three hidden layers, where the number of LSTM units in each layer is set to 50, 100 and 50. All the benchmark models are compiled with the SGD learning algorithm whilst the CURNet benefits from the RGD learning algorithm. A mini-batch size of 1000 samples is used to accelerate the

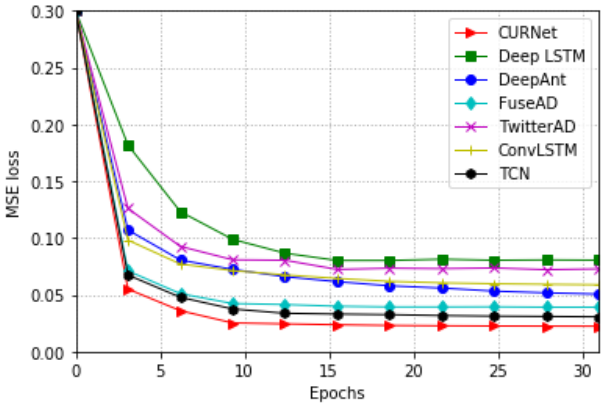


FIGURE 5. Autoregression MSE loss comparison of different models with $K = 8$ and $L = 6$.

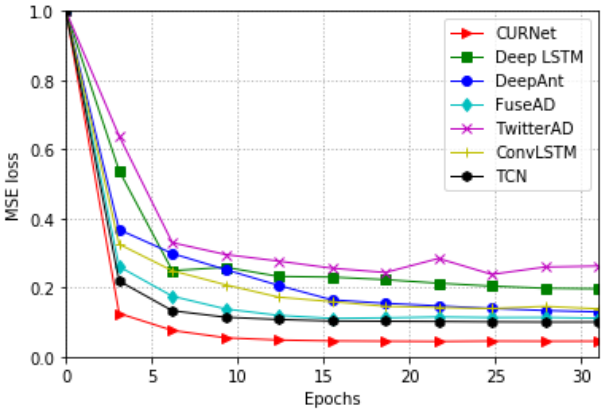


FIGURE 6. Autoregression MSE loss comparison of different models with $K = 16$ and $L = 12$.

training process for all models. Fig. 5 illustrates the mean squared error (MSE) performance with respect to the number of epochs on the testing data. The proposed recursive deep LSTM architecture is able to outperform the compared state-of-the-art fault prediction techniques.

Secondly, we investigate these techniques in a more challenging scenario by setting the number of lookbacks $K = 16$ and $L = 12$, which will give 2 minutes for advanced fault prevention. For all the compared networks, the number of units and network hyperparameters remain the same as in the previous setting. The simulation results are as illustrated in Fig. 6. With a larger number of lookbacks, the proposed CURNet is still able to outperform the other compared networks significantly, albeit all techniques have some performance losses comparing to the previous case. While the other techniques still appear to suffer from performance degradation as the L increases, the proposed CURNet is able to maintain better robustness against the uncertainty accumulations over extended time steps.

Thirdly, the proposed RGD learning algorithm is evaluated comparing with the conventional SGD applied for the CURNet and the result is as shown in Fig. 7. The value of input data

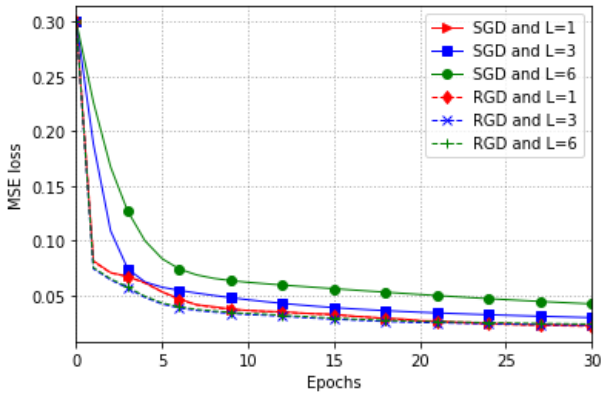


FIGURE 7. Autoregression MSE loss comparison of RGD and SGD learning algorithms with different L values.

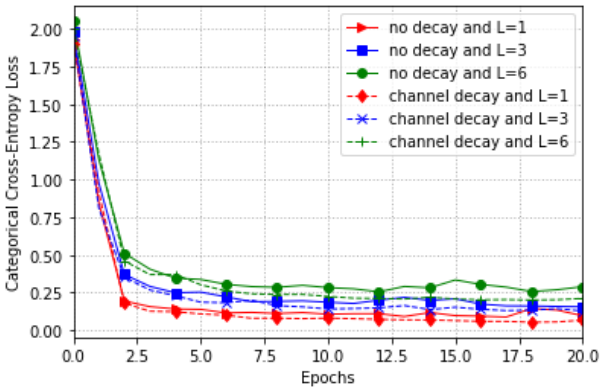


FIGURE 8. Fault classification categorical cross-entropy loss comparison of CURNet with/without channel decay effect using different L values.

lookbacks is fixed at $K = 8$ while we examine the learning performance in terms of different L values. Additionally, we include single time step prediction (i.e., $L = 1$) as a special case. The rest network hyperparameters are kept the same as in the previous simulations. In Fig. 7, the solid and dashed lines are for the SGD and RGD algorithms, respectively. As can be seen, the proposed RGD algorithm helps to achieve lower MSE than the SGD in general, and the MSE differences between the cases with different choices of L values are unnoticeable. As aforementioned, the RGD learning algorithm learns to reduce the cumulative uncertainties over multiple recursive predictions.

Moreover, the effect of using channel decay is evaluated in Fig. 8, in which the solid lines represent the case where channel decays are not considered whereas the dashed lines represent the case where channel decays are implemented. In the latter case, we keep the channel predicting parameter $p = 0.95$ (p is reasonable to be set closer to 1 when L is large enough, so that predicting factor for the Lth step p^{L-1} is not too small to be ignored). For the network hyperparameters, we use 10 filters with a kernel size of 5 for the first 1D convolution layer, followed by a batch normalization and a ReLu activation function, after which an upsampling layer with sampling rate of 0.5 is applied. Subsequently, another set of temporal convolution layer with batch normalization, Relu and upsampling of the same values is added. Further, the output of the previous layers is flattened before being fed in two fully-connected dense layers of 30 and 4 neurons, respectively. Finally, a softmax activation function is implemented before the output layer to compute the categorical cross-entropy loss function. The 4 neurons in the output layer indicate the 4 sensors which are previously injected with faults, in convenience to identify different fault types and locations. It can be observed that the categorical cross-entropy loss increases as the number of prediction step L increases. However, the network is able to achieve a lower loss when the channel decay effect is also implemented.

Finally, the precision-recall area under the curve (PR AUC) of the compared models are illustrated in Fig. 9 In order

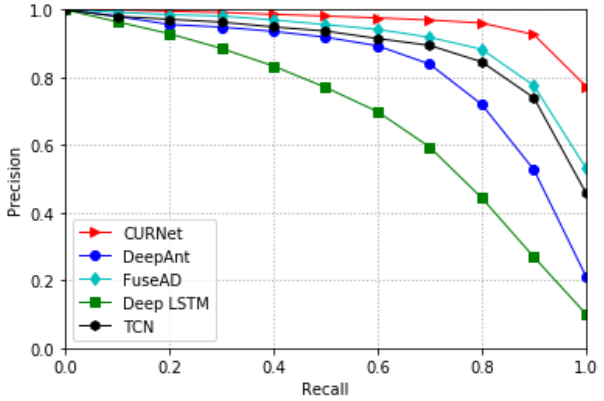


FIGURE 9. PR AUC comparison of different models with $K = 8$ and $L = 6$.

to facilitate this comparison, we consider the CURNet as a binary classifier by disregarding the classes of faults it classifies. As long as a fault is predicted correctly, it is categorized as true positive. We keep $K = 8$, $L = 6$ and $p = 0.95$ while leaving the other hyperparameters for the selected models remain the same as in the previous cases. From Fig. 9, it is obvious the CURNet outperforms the other techniques in classifying faults from imbalanced data. This is because that the superiority of CURNet pertaining to fault classification is contributed by not only the accurate autoregression of phase one, but also the implementation of channel decay effect defined by the predicting factor for the convolutional process to assign more weights on the more immediate prediction steps and less weights on the further steps, which is an effective way to model the confidence level at different time steps. This equips CURNet with great capability in predicting and spotting out rare faults with high precision in a fault-sensitivity manner.

The hyperparameters of CURNet are elaborated in Table 2, which summarizes all the simulation cases used to produce results illustrated in Fig. 5-9 in this section. We would like to remark that the LSTM units combination listed (i.e., $50 - 100 - 50$) is the optimal choice for producing the most

TABLE 2. CURNet hyperparameters.

Scenarios	Phase one: LSTM units in layers	Phase two: filters, filter size & dense layer, neurons	Lookbacks K	Prediction steps L	Predicting factor p	Learning algorithm & learning rate	Seconds predicted/earned to prevent faults (secs)
Case 1	$50-100-50$	-	8	6	-	RGD, 0.01	60
Case 2	$50-100-50$	-	16	12	-	RGD, 0.01	120
Case 3	$50-100-50$	$10,(5,)$ $30-4$	8	6	0.95	SGD & RGD, 0.01	60
Case 4	-	$10,(5,)$ $30-4$	-	6	1 & 0.95	RGD, 0.01	60
Case 5	$50-100-50$	$10,(5,)$ $30-4$	8	6	0.95	RGD, 0.01	60

TABLE 3. Complexity comparison.

Deep networks	Batch size	Epochs	Complexity in training time (mins)	Response time for making a prediction (secs)
CURNet	1000	30	47	0.2734
Deep LSTM	1000	50	107	0.5524
DeepAnT	1000	40	42	0.0760
FuseAD	1000	50	133	0.3604
TwitterAD	1000	30	38	0.0588
ConvLSTM	1000	40	57	0.2981
TCN	1000	30	28	1.1421

consistent and satisfactory results whilst not overburdening the computational complexity, compared to other tested combinations (i.e., $30-60-30$ and $60-120-60$). In addition, the complexity of all the compared algorithms is shown in Table 3. Deep learning network complexity is often reflected by both the training time and the real-time responses to make a prediction. In order to ensure fairness, the batch size for training is fixed at 1000 samples per batch for all models. The average epochs required for training to converge also varies depending on the selected model, which is presented in Table 3. Both training time and response time for making a prediction are used to reflect the actual run-time (i.e., real-time) complexity for all compared models. It is clear that CURNet is relatively efficient at training comparing to the other models. It also provides a very short response time / latency (i.e., 0.2734 seconds) to making an online prediction for the machine learning module of the CPS to minimize the overall processing delays.

VI. CONCLUSION

In order to deal with challenging fault prediction problems in WSN embedded industrial CPS, we have developed a novel and effective end-to-end deep learning network named CURNet with its own novel learning algorithm named RGD. The proposed CURNet has shown superior performance as compared with the state-of-the-art time-series fault prediction models in terms of both fault prediction recall and fault type classification accuracy.

For future work, first of all, the novel RGD learning algorithm can be tested for other recursive deep networks with online prediction capacities for similar industrial process applications, since it has only been verified to be working sufficiently well for the proposed CURNet in this work. Secondly, it is also important to test the CURNet on an even larger dataset to verify its statistical robustness for big data IIoT environments and how its complexity scales up with data size. A larger dataset usually contains more heterogeneous types of faults and some faults may appear more often than others, which brings further challenges to classification tasks. Thirdly, it would be interesting to find out if CURNet works on a balanced fault-ratio dataset. Although this is not the case for industrial processes, it is worthwhile to compare to existing models in terms of general time-series prediction performance with different error metrics. Lastly, since the complexity of CURNet increases with the dimension of input features, it may be helpful to develop an efficient feature encoding method to reduce the feature dimensions of the chemical industrial process without affecting the variable dependencies.

REFERENCES

[1] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, Apr. 2014.

[2] L. An and G.-H. Yang, "Byzantine-resilient distributed state estimation: A min-switching approach," *Automatica*, vol. 129, Jul. 2021, Art. no. 109664.

[3] L. An and G.-H. Yang, "Distributed secure state estimation for cyber–physical systems under sensor attacks," *Automatica*, vol. 107, pp. 526–538, Sep. 2019.

[4] A.-Y. Lu and G.-H. Yang, "Secure state estimation for multiagent systems with faulty and malicious agents," *IEEE Trans. Autom. Control*, vol. 65, no. 8, pp. 3471–3485, Aug. 2020.

[5] C. Kwon and I. Hwang, "Reachability analysis for safety assurance of cyber-physical systems against cyber attacks," *IEEE Trans. Autom. Control*, vol. 63, no. 7, pp. 2272–2279, Jul. 2018.

[6] P. J. Bonczek and N. Bezzo, "Detection of hidden attacks on cyber-physical systems from serial magnitude and sign randomness inconsistencies," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 3281–3287.

[7] B. Croteau, D. Krishnankutty, R. Robucci, C. Patel, N. Banerjee, K. Kiriakidis, T. Severson, and E. Rodriguez-Seda, "Cross-level detection of sensor-based deception attacks on cyber-physical systems," in *Proc. IEEE 7th Annu. Int. Conf. CYBER Technol. Autom., Control, Intell. Syst. (CYBER)*, Jul. 2017, pp. 1037–1042.

[8] D. Ye and T.-Y. Zhang, "Summation detector for false data-injection attack in cyber-physical systems," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2338–2345, Jun. 2020.

[9] M.-Q. Tran, M. Elsisi, K. Mahmoud, M.-K. Liu, M. Lehtonen, and M. M. F. Darwish, "Experimental setup for online fault diagnosis of induction machines via promising IoT and machine learning: Towards industry 4.0 empowerment," *IEEE Access*, vol. 9, pp. 115429–115441, 2021.

[10] I. Koren, "Detecting and counteracting benign faults and malicious attacks in cyber physical systems," in *Proc. 7th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2018, p. 2.

[11] L. Wen, X. Li, L. Gao, and Y. Zhang, "A new convolutional neural network-based data-driven fault diagnosis method," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5990–5998, Jul. 2018.

[12] W. Ni, K. Wang, T. Chen, W. J. Ng, and S. K. Tan, "GPR model with signal preprocessing and bias update for dynamic processes modeling," *Control Eng. Pract.*, vol. 20, no. 12, pp. 1281–1292, Dec. 2012.

[13] K. Wang, L. Zhuo, Y. Shao, D. Yue, and K. F. Tsang, "Toward distributed data processing on intelligent leak-points prediction in petrochemical industries," *IEEE Trans. Ind. Informat.*, vol. 12, no. 6, pp. 2091–2102, Dec. 2016.

[14] Z. Gao, L. Ma, and J. Wang, "Fault tolerant control method for displacement sensor fault of wheel-legged robot based on deep learning," in *Proc. WRC Symp. Adv. Robot. Autom. (WRC SARA)*, Aug. 2018, pp. 147–152.

[15] F. Caliva, F. S. De Ribeiro, A. Mylonakis, C. Demazirere, P. Vinai, G. Leontidis, and S. Kollias, "A deep learning approach to anomaly detection in nuclear reactors," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[16] B. Bhadriraju, J. S.-I. Kwon, and F. Khan, "Dynamic risk-based fault prediction of chemical processes using online sparse model identification," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 4964–4969.

[17] M. S. Parwez, D. Rawat, and M. Garuba, "Big data analytics for user-activity analysis and user-anomaly detection in mobile wireless network," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2058–2065, Aug. 2017.

[18] A. C. Onal, O. B. Sezer, M. Ozbayoglu, and E. Dogdu, "MIS-IoT: Modular intelligent server based Internet of Things framework with big data and machine learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2270–2279.

[19] C.-F. Lai, W.-C. Chien, L. T. Yang, and W. Qiang, "LSTM and edge computing for big data feature recognition of industrial electrical equipment," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2469–2477, Apr. 2019.

[20] S. Rashid, U. Akram, S. Qaisar, S. A. Khan, and E. Felemban, "Wireless sensor network for distributed event detection based on machine learning," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom)*, Sep. 2014, pp. 540–545.

[21] H. Martins, F. Januário, L. Palma, A. Cardoso, and P. Gil, "A machine learning technique in a multi-agent framework for online outliers detection in wireless sensor networks," in *Proc. 41st Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2015, pp. 000688–000693.

[22] J. Kwak, T. Lee, and C. O. Kim, "An incremental clustering-based fault detection algorithm for class-imbalanced process data," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 3, pp. 318–328, Aug. 2015.

[23] H.-C. Yan, J.-H. Zhou, and C. K. Pang, "New types of faults detection and diagnosis using a mixed soft & hard clustering framework," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2016, pp. 1–6.

[24] H. Ruan, B. Dorneanu, A. Mohamed, M. Abdelwahab, Y. Gao, H. Arellano-Garcia, and P. Xiao, "Towards fault detection and self-healing of chemical processes over wireless sensor networks," in *Industry 4.0-Shaping The Future of The Digital World*. New York, NY, USA: Taylor & Francis, Apr. 2019.

[25] A. Mohamed, H. Ruan, M. H. H. Abdelwahab, B. Dorneanu, P. Xiao, H. Arellano-Garcia, Y. Gao, and R. Tafazolli, "An inter-disciplinary modelling approach in industrial 5G/6G and machine learning era," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[26] L. Yunpeng, H. Di, B. Junpeng, and Q. Yong, "Multi-step ahead time series forecasting for different data patterns based on LSTM recurrent neural network," in *Proc. 14th Web Inf. Syst. Appl. Conf. (WISA)*, Nov. 2017, pp. 305–310.

[27] Y. Wang, S. Zhu, and C. Li, "Research on multistep time series prediction based on LSTM," in *Proc. 3rd Int. Conf. Electron. Inf. Technol. Comput. Eng. (EITCE)*, Oct. 2019, pp. 1155–1159.

[28] D. Sahoo, N. Sood, U. Rani, G. Abraham, V. Dutt, and A. D. Dileep, "Comparative analysis of multi-step time-series forecasting for network load dataset," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–7.

[29] H. Rodriguez, M. Medrano, L. M. Rosales, G. P. Penunuri, and J. J. Flores, "Multi-step forecasting strategies for wind speed time series," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, vol. 4, Nov. 2020, pp. 1–6.

[30] X. Wang and Y. Zhang, "Multi-step-ahead time series prediction method with stacking LSTM neural network," in *Proc. 3rd Int. Conf. Artif. Intell. Big Data (ICAIBD)*, May 2020, pp. 51–55.

[31] B. Lindemann, N. Jazdi, and M. Weyrich, "Anomaly detection and prediction in discrete manufacturing based on cooperative LSTM networks," in *Proc. IEEE 16th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2020, pp. 1003–1010.

[32] S. Fan, X. Zhang, and Z. Song, "Imbalanced sample selection with deep reinforcement learning for fault diagnosis," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2518–2527, Apr. 2022.

[33] I. Pisa, I. Santin, J. L. Vicario, A. Morell, and R. Vilanova, "Data preprocessing for ANN-based industrial time-series forecasting with imbalanced data," in *Proc. 27th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2019, pp. 1–5.

[34] Y. Fathy, M. Jaber, and A. Brintrup, "Learning with imbalanced data in smart manufacturing: A comparative analysis," *IEEE Access*, vol. 9, pp. 2734–2757, 2021.

[35] M. David Dangut, Z. Skaf, and I. Jennions, "Rescaled-LSTM for predicting aircraft component replacement under imbalanced dataset constraint," in *Proc. Adv. Sci. Eng. Technol. Int. Conf. (ASET)*, Feb. 2020, pp. 1–9.

[36] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019.

[37] M. Munir, S. A. Siddiqui, M. A. Chattha, A. Dengel, and S. Ahmed, "FuseAD: Unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models," *Sensors*, vol. 19, no. 11, p. 2451, May 2019.

[38] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, "Automatic anomaly detection in the cloud via statistical learning," 2017, *arXiv:1704.07706*.

[39] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. 33rd AAAI Conf. Artif. Intell. (AAAI)*, Nov. 2019, pp. 1409–1416.

[40] H. Ruan and R. C. de Lamare, "Distributed robust beamforming based on low-rank and cross-correlation techniques: Design and analysis," *IEEE Trans. Signal Process.*, vol. 67, no. 24, pp. 6411–6423, Dec. 2019.

[41] A. Abid, M. Khan, and J. Iqbal, "A review on fault detection and diagnosis techniques: Basics and beyond," *Artif. Intell. Rev.*, vol. 54, no. 5, pp. 3639–3664, Jun. 2021.

[42] Y. Wang, Y. Si, B. Huang, and Z. Lou, "Survey on the theoretical research and engineering applications of multivariate statistics process monitoring algorithms: 2008–2017," *Can. J. Chem. Eng.*, vol. 96, no. 10, pp. 2073–2085, Oct. 2018.

[43] X. Zhang, M. Kano, M. Tani, J. Mori, J. Ise, and K. Harada, "Poisson mixture model for defects prediction in steelmaking," in *Proc. 58th Annu. Conf. Soc. Instrum. Control Eng. Jpn. (SICE)*, Sep. 2019, pp. 1274–1279.

[44] X. Yuan, L. Li, and Y. Wang, "Nonlinear dynamic soft sensor modeling with supervised long short-term memory network," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3168–3176, May 2020.

[45] Y. Pan, Q. Li, H. Liang, and H.-K. Lam, "A novel mixed control approach for fuzzy systems via membership functions online learning policy," *IEEE Trans. Fuzzy Syst.*, early access, Nov. 24, 2021, doi: 10.1109/TFUZZ.2021.3130201.

[46] H. Ruan, P. Xiao, L. Xiao, and J. R. Kelly, "Joint iterative optimization-based low-complexity adaptive hybrid beamforming for massive MU-MIMO systems," *IEEE Trans. Commun.*, vol. 69, no. 3, pp. 1707–1722, Mar. 2021.

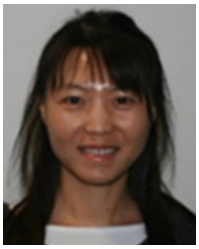
[47] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1003–1012.

HANG RUAN (Member, IEEE) received the B.Eng. degree from the University of Liverpool, U.K., in 2011, and the M.Sc. degree (Hons.) in digital signal processing and the Ph.D. degree in sensor array signal processing from the University of York, in 2012 and 2017, respectively. From 2018 to 2019, he worked as a Research Fellow on deep learning with the 5G Innovation Center (5GIC, now 6GIC), University of Surrey. Since 2021, he has been working as a Research Associate on machine learning for data science with the School of Mathematics, University of Edinburgh. His research interests include cross-disciplines in machine learning, computational optimization, mathematics, statistical and adaptive signal processing, beamforming, sensor networks, data analytics, and predictive modeling with applications in artificial intelligence, electrical and electronic engineering, and data science.

BOGDAN DORNEANU is currently a Postdoctoral Researcher with the Department of Process and Plant Technology, Brandenburg University of Technology (BTU), leading the Digitalization Group. His research interests include process and product modeling and simulation, model predictive control, and optimization and design of systems and processes, including models and decision-making tools for distributed energy resource networks.

HARVEY ARELLANO-GARCIA received the Ph.D. degree in process systems engineering with a focus on process intensification and optimization from the TU Berlin, Germany. He studied energy and process engineering at the TU Berlin. He founded and headed the ''Process and Energy Systems Engineering'' Research Group, TU Berlin, from 2007 to 2012. During this time, he carried out research stays at Imperial College London and MIT, Boston. He is currently the Head of the Department of Process and Plant Technology, Brandenburg University of Technology, Germany, and a Visiting Professor at the University of Surrey, U.K. From 2012 to 2019, he was appointed as a University Professor, U.K., and the Research Director and a Professor of chemical engineering at the University of Surrey. He has extensive experience in planning, launching, and managing multidisciplinary research projects with a strong track record of successful tech transfer to the industry. He is in various national and international bodies and organizations in the field of process engineering and control technology. He has published more than 180 peer-reviewed publications, including four patents. His research interests include the application of mathematical methods to optimize process design, control, and operation, as well as model-based experimental analysis in process and energy systems. His work has been awarded various prizes, including the European Federation of Chemical Engineering (EFCE) Excellence Award in recognition of his outstanding Ph.D. in Computer-Aided Process Engineering (CAPE). He has been a PI in several academia-industry and research projects. These research results have also been successfully integrated into several industrial processes.

PEI XIAO (Senior Member, IEEE) is currently a Professor of wireless communications at the Institute for Communication Systems, home of 5GIC and 6GIC, University of Surrey. He is also the Technical Manager of 5GIC/6GIC, leading the Research Team in the new physical layer work area, and coordinating/supervising research activities across all the work areas (https://www.surrey.ac.uk/institute-communication-systems/5g-6g-innovation-centre). Prior to this, he worked at Newcastle University and Queen's University Belfast. He also held positions at Nokia Networks, Finland. He has published extensively in the fields of communication theory, RF and antenna design, and signal processing for wireless communications. He is an inventor on over ten recent 5GIC patents addressing bottleneck problems in 5G systems.



LI ZHANG (Senior Member, IEEE) received the Ph.D. degree from the University of Birmingham. She gained postdoctoral experience from the University of Birmingham. She is currently a Reader with the Department of Computer Science, Royal Holloway, University of London, U.K. Her research interests include deep learning, machine learning, intelligent robotics, and evolutionary computation. She is an Associate Editor of *Decision Support Systems*.