A LSTM-based Service Failure Prediction Approach for Predictive Maintnenace in IoT Systems

Pan He, Yue Yuan, Gang Liu
Chongqing Institute of Green and Intelligent Technology,
Chinese Academy of Sciences
Chongqing, China
e-mail: hepan@cigit.ac.cn

***Abstract*—To reduce the costly impact caused by downtime in IoT systems, proactive or predictive maintenance is employed as the main approach. In the whole process, the time to failure should be predicted in advance to estimate the optimum maintenance time. Existing research on online failure prediction generally concentrates on state changing trend prediction or system-level prediction. With a massive number of underlying services, the system-level prediction could not indicate the specific component to fail. Furthermore, user-defined threshold or arbitrary rules in existing approaches may cause wrong decisions when determining whether the failure is about to occur in a certain time based on the state changing trend. To predict whether some service is about to fail in future, this paper proposes a multi-layer data-driven failure prediction model for predictive maintenance. The failure prediction problem is cast into a time-lagged multi-class classification problem. The time-series quality data, historical error and maintenance data are all used as the model features while the time-lagged individual service failure information is employed as the class label. A multi-layer LSTM based algorithm is used to solve this classification problem. The prediction result includes both the remaining useful life time and the specific service to fail. The prediction method is experimented on an open source IoT maintenance dataset and compared with some existing machine learning methods. The comparison results show that without the additional feature engineering work, the LSTM based approach outperforms most of the traditional machine learning methods in both precision and recall metrics.**

Keywords-online failure prediction; long short-term memory; IoT system; multi-class classification; predictive maintenance

I. INTRODUCTION

While the physical products become software defined nowadays, the Internet of Things (IoT) technology establishes more embedded connectivity and diligent data management [1]. According to Gartner, the number of connected devices has been growing exponentially and is expected to grow to 25 billion by 2021 [2]. A major problem faced in an asset-heavy IoT application is the significant costs associated with delays due to unpredictable device failures. To reduce the costly impact caused by downtime, proactive or predictive approaches predicts system failures in advance so that issues could be proactively fixed before occurring [3].

In service-oriented systems, proactive maintenance is carried out by the service composition self-adaptation, before the service quality degrades below the user's requirement [4-6]. During the proactive maintenance process, online failure prediction plays an important role in computing systems [7]. It prevents the occurrence of failure by predicting time to failure and estimating the optimum time to do maintenance [8, 9]. Although the problem of online failure prediction has been studied for years [7, 10, 11], the real merge between physical world and digital world introduces new challenges. First of all, the primary concern in IoT applications would be to make a precise decision on the time to trigger maintenance, instead of predicting the service changing trend. Secondly, for complex systems executing in dynamic environment, state prediction highly relies on the time series historical data instead of discrete data in certain time [12, 13]. Thirdly, the individual service in an IoT application could be fixed separately. It would be more efficient to handle the potential failure in IoT applications, if the failure service could be stated clearly in advance. In a word, for IoT applications, the service to fail should be predicted and a decision to maintain should be made in advance, using the time-series monitoring data.

Existing work on time series prediction in service-oriented systems generally concentrates on service quality degrading trend prediction or the remaining useful life (RUL) time prediction. However, there are few mature methods to choose the criteria, which determines whether the failure is about to occur to the specific service or whether an adaption should be triggered. On one hand, regression, time-series or machine learning methods are generally used to build time-aware quality prediction models for individual service [12, 14, 15]. Rule-based methods or use-defined threshold values are used to trigger the adaptation or maintenance process. However, it is not feasible to monitor each underlying device and to choose the corresponding adaptation threshold. The rule-based methods are too arbitrary in dynamically changing execution environment and may cause wrong decisions. As a result, the criteria or the method to indicate the service failure through the time-related quality data should be further researched. On the other hand, RUL prediction helps to determine the time to failure of a system by regression or classification models. The prediction is conducted using the general monitoring information collected at the system or sub-system level. However, the specific service to fail or the failure type could not be specified using the time-series prediction model. With a massive number of underlying services, it might be difficult to conduct failure diagnosis and

service identification. Although some attentions have been paid to identify the component to fail in predictive maintenance, the traditional machine learning method is employed which only fit for non-time aware features [16]. The time-series information is transformed into discreate data in system-unique ways, which is not applicable for other systems and may cause low prediction accuracy.

To solve the failure component identification problem in IoT systems, this paper presents a multi-layer data-driven failure prediction model for proactive maintenance. The service failure prediction problem is cast into a time-lagged multi-class classification problem. During the model construction process, the time-series quality data, historical error and maintenance data are all used as the model features while the time-lagged service failure information is employed as the class label. The prediction result, as well as the class label, indicates both the remaining useful life time and the specific service to fail. To avoid feature selection and construction complexity, a type of recurrent neural network (RNN), the long short-time memory (LSTM), is used to solve the model.

The main contributions of the paper are summarized as follows:

- The problem of when and where a service-oriented system is going to fail, is further researched in this paper, based on the time series historical data.
- A multi-layer LSTM based approach is used to solve the time-lagged multi-class classification problem, to improve the classification accuracy.
- The prediction method is experimented on an open source IoT maintenance dataset and compared with some existing machine learning methods, to prove its efficiency.

The rest of the paper is organized as follows. Section II discussed the related work on predictive models. The multi-class prediction problem is formulated in section III. The proposed time-lagged classification method is proposed in section IV. Experimental results and analysis are listed in section V followed by the conclusions in section VI.

II. Related Work

The problem of proactive reliability assurance has already attracted attentions from many researchers. Existing works mainly concentrate on online prediction.

A. Service-oriented System

Traditionally, service adaptation is triggered reactively when a service becomes unavailable during execution [17]. To address the drawbacks of reactive adaptation, preventive adaptation monitors the service deviation, and triggers adaption before the deviation occurs [5, 18]. Since monitoring only observes changes or deviations after they have occurred, faulty services may also cause unwanted consequences. To detect service degradation in advance, predictive approaches are the main choices in existing work.

For the support of the proactive decisions, forecasting approaches are developed to obtain the future values of dynamic QoS attributes [19]. Regression methods [14], Markov-related models [20] and ARIMA models [12] are all used for time series quality data prediction. Using statistical time series models, each subsequent prediction depends on the previous predicted result and it is difficult to assure high prediction accuracy over multiple time steps. So, the time series modelling approaches are further improved by RNN, LSTM, or dynamic Bayesian networks. To improve the prediction accuracy, Wang et al. employed motifs-based dynamic Bayesian networks and LSTM network to estimate the web service's reliability in the near future [13, 21]. Xiong et al. also proposed a LSTM based matrix factorization approach for online QoS prediction [22].

The above research only evaluates future quality of individual service, further studies are needed to determine whether to trigger dynamic service adaptation based on the prediction results. Using an exponentially weighted moving average model for aggregated QoS value prediction, Aschoff et al. performed the verification of service-level agreement (SLA) violation using the SLA requirement as the threshold value [4, 23]. Based on the service forecasting using the least squares support vector machine, Hu et al. proposed a proactive service selection approach, employing a certain threshold value to verify the predicted value and to trigger the negation-based service cosigning process [10]. Taking into account the prediction error between the predicted value and the real value, the mean prediction error and the target SLO value were both used in the threshold to trigger adaptations [24]. With the help of the online time series prediction model, Wang et al. proposed a proactive approach to prevent the occurrence of failure with the comparison between QoS constraints [6]. In the above approaches, the service adaptation decision was made through the comparison between the predicted values, the user constraints and the threshold values. While the threshold values are important to trigger adaptation process, they are presented in an arbitrary way without an exact method. Besides, an adaptation decision is made through comparisons. In dynamic changing environment, the simple comparison strategy of whether a quality value exceeds the threshold may cause wrong decisions.

B. IoT Based Industrial System

For IoT environment, predictive analytics algorithms are used to dynamically manage proactive maintenance policies [3]. Instead of service quality, it mainly focuses on the RUL time of a system. The existing research often assumes a stochastic degradation process and an aperiodic or periodic inspection plan to trigger the preventive or corrective replacement [8]. Omshi et al. formulated the RUL time by the distribution of the degradation process and the maintenance plan was triggered by the *p*-quantile of the RUL distribution [25]. Based on the stochastic assumption, Ruiz-Sarmiento et al. used a predictive model based on a Discrete Bayesian Filter, to estimate and predict the gradual degradation of such machinery, permitting informed decisions regarding maintenance operations [26].

Due to the dynamic execution environment, the degradation process may not obey certain stochastic features. As a result, data-driven methods have become the most effective solutions to address fault diagnosis and remaining life assessment, with the tremendous revival of artificial intelligence [9, 11]. Zhang et al. used a PCA based feature

extraction method and a LSTM to develop a lightweight predictive maintenance model [27]. They formalized the failure prediction problem as a binary classification problem to indicate whether the system is going to fail in a certain amount of time. Nguyen et al. formulated the similar problem into a multi-class classification problem with numerous RUL time windows [28]. The classification result indicated whether the system would fail in different time windows. The above research only generates the result of RUL time while the service to maintain should be further investigated. Microsoft Azure AI Lab had paid attention to distinguish the component to fail in predictive maintenance through multi-class classification [16]. However, the time-series information was transformed into discrete data and traditional machine learning methods were used. This kind of approach could not be widely used since the transformation technique was unique for each system and inappropriate transformation would lead to low prediction accuracy. Instead, original time series data needs to be used in the multi-class classification.

III. PROBLEM FORMULATION

In this section, the time-lagged multi-class classification problem to solve in this paper is presented.

A. Time-lagged Feature Prediction

In IoT systems, such as a power plant, sensors are deployed to monitor system status at real time. As different sensors concentrate on different types of data, the monitored data is referred to as "features" for general description. Certain number of features are collected by one or multiple sensors at certain time interval. At each observation time point, each collected sensor data is represented as a tuple of the feature value with the timestamp, such as $<t_i^j, q_i^j>$. q_i^j denotes the *j*-th observation of the *i*-th feature collected from a sensor and t_i^j denotes the corresponding timestamp.

While multiple features are collected simultaneously, it is assumed that each feature from different sensors is collected at the same time interval, that is $t_i^j = t_{i+1}^j$ and $t_i^j - t_i^{j-1} = t_i^{j+1} - t_i^j (j > 0)$. For simplicity, let t^j denote the time to collect the *j*-th observations for all features. The features collected at time t^j is represented as $< q_1^j, q_2^j, q_3^j \dots q_m^j >$, where *m* denotes the number of all the features.

After collecting data for a certain amount of time, continuous data is collected data for each kind of feature with timestamps. Time series is a common approach to formulate the data for further analysis. Let $\boldsymbol{q}_i^{j,h}$ denote a segmented time series of the *i*-th feature, collected from time t^j to t^h, with $h - j + 1$ observations. $\boldsymbol{q}_i^{j,h}$ is represented in a vector form as $\boldsymbol{q}_i^{j,h} = (q_i^j, q_i^{j+1}, q_i^{j+2}, \dots, q_i^h)$. The key problem for the online prediction in the services-oriented system is to predict the system status q_i^k in a future time $t^k (t^k > t^h)$ according to the time series feature values $\boldsymbol{q}_i^{j,h}$ in the past. As shown in Fig.1, Δt_h represents the recent historical time period, during which $h - j + 1$ observations are collected for each feature, $\Delta t_h = t^h - t^j$. Δt_k is the effective prediction time period $\Delta t_k = t^k - t^h$. $\boldsymbol{q}_i$ denotes the continuous time series data.

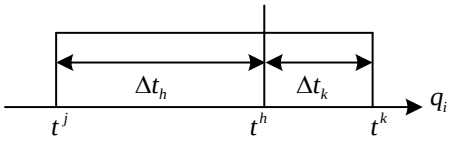

Figure 1. The time-lagged feature prediction.

Similarly, Let $\boldsymbol{q}^j = (q_1^j, q_2^j, q_3^j \dots q_m^j)^T$ denote the vector describing multiple feature values at time t^j. $\boldsymbol{q}^{j,h}$ represents the segmented multivariate time series, collected from time t^j to t^h. $\boldsymbol{q}^{j,h}$ is represented in a matrix form as $\boldsymbol{q}^{j,h} = (\boldsymbol{q}_1^{j,h}, \boldsymbol{q}_2^{j,h}, \boldsymbol{q}_3^{j,h} \dots \boldsymbol{q}_m^{j,h})^T$. Additional feature values, such as RUL time in IoT systems, could be transformed from the original observed feature data and included in these vectors. The multivariate time-lagged prediction problem is defined as predicting the vector $\hat{\boldsymbol{q}}^k$ containing the predicted feature values for a future time point t^k from the multivariate historical time series $\boldsymbol{q}^{j,h}$

$$\hat{\boldsymbol{q}}^k = \boldsymbol{f}(\boldsymbol{q}_1^{j,h}, \boldsymbol{q}_2^{j,h}, \boldsymbol{q}_3^{j,h} \dots \boldsymbol{q}_m^{j,h}), \quad (1)$$

where $\boldsymbol{f}$ denotes the multivariate prediction model.

B. Time-lagged Multi-Class Classification

The above model only describes the problem of predicting the time series feature value in future. Another problem exists in the IoT system is to determine whether a system fails in a period of time according to the feature values in the past. It generates a more straightforward result than the RUL time prediction. The above problem could be formulated into a classification problem based on the time series feature values. Two examples are presented in the existing research. Given a RUL time window Δt_k, the conclusion of whether the system fails during Δt_k, is comprised of two labels: yes or no [27]. In this case, the two labels are used for binary classification. If multiple RUL time windows as given for failure prediction, as shown in Fig. 2, the conclusion should be made on whether the system fails during Δt_{k1} or Δt_{k2}. The results are then divided into three classes: the system fails during Δt_{k1}, the system fails during Δt_{k2} and the system does not fail during neither of the time windows. The problem is then cast into a 3-class classification problem [28].

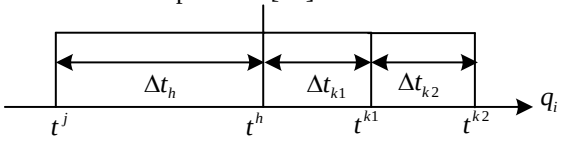

Figure 2. The classification with multiple time windows.

The above problems only concentrate on the RUL time estimation and the system-level failure prediction. The problem to solve in this paper is which service is going to fail during a given time window Δt_k. It is assumed that there are *s* services which may fail in the system. Let $l_w (0 \le w \le s)$ represent the result that the *w*-th service will fail within the time window Δt_k. The condition that $w = 0$ indicates that none of the services will fail. Following the above

assumptions, let v^k denote the classification result at time t^k, with a given prediction period Δt_k. The main concern of the classification problem is to evaluate the probability of each class labels l_i at a future time t^k, with the given data: $p_w^k = \Pr(v^k = l_w|\boldsymbol{q}^{j,h})$. Let $\boldsymbol{p}^k$ denote the vector containing the probabilities of each class at a future time point $\boldsymbol{p}^k = (p_0^k, p_1^k, \dots p_s^k)$. The time-lagged multi-class classification problem is defined as predicting the value of $\boldsymbol{p}^k$ according to the historical time series $\boldsymbol{q}^{j,h}$

$$\hat{\boldsymbol{p}}^k = \boldsymbol{g}\left(\boldsymbol{q}_1^{j,h}, \boldsymbol{q}_2^{j,h}, \boldsymbol{q}_3^{j,h} \dots \boldsymbol{q}_m^{j,h}\right), \tag{2}$$

where $\hat{\boldsymbol{p}}^k = (\hat{p}_0^k, \hat{p}_1^k, \dots \hat{p}_s^k)^T$ denotes the predicted probability of $\boldsymbol{p}^k$ and $\boldsymbol{g}$ denotes the classification model. After evaluating the probability for each class, the label of the class with the maximal probability is chosen as the final output

$$v^k = l_w^k, \text{where } \hat{p}_w^k = \max(\hat{\boldsymbol{p}}^k). \tag{3}$$

So, the main work of the rest part of this paper is to find the appropriate method to solve $\boldsymbol{g}$.

IV. A LSTM-BASED CLASSIFICATOIN METHOD

The above problem is a multivariate multi-class classification problem. To improve the classification accuracy, a multi-layer deep learning method is utilized in this section.

A. Feature Segmentation

To build the time series classification model, continuous time series feature values are segmented the model input. The main work in the data pre-processing step is to generate fixed-length time series from for each feature. Assuming that the time steps of each time series sequence is *n*, the set of the segmented time series for the *i*-th feature is $\boldsymbol{Q_i} = (\boldsymbol{q}_i^{1,n}, \boldsymbol{q}_i^{2,n+1}, \boldsymbol{q}_i^{3,n+2}, \dots \boldsymbol{q}_i^{j,n+j-1}, \dots)$, where $\boldsymbol{q}_i^{1,n} = (q_i^1, q_i^2, q_i^3, \dots, q_i^n)$. For multiple features, the long-term time series are all split into fixed-length sequences. The segmented multivariate time series could be represented as 3-dimensional array $\boldsymbol{Q} = (\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{Q}_3 \dots \boldsymbol{Q}_m)^T$, with the size of $m \times n \times x$, as shown in Fig. 3. *x* is the number of time series sequences; *n* is the look back window or sequence length and *m* is the number of features of each sequence at each time step.

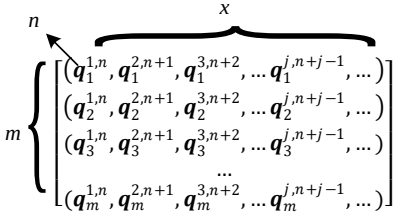

Figure 3. An example of the segmented time serise set.

B. Multivariate LSTM Network

In a dynamic execution environment, the feature time series may not follow the stationary pattern. Thus, nonlinear patterns and nonlinear relations should be handled during the time series forecasting [29]. As new-emerging technology, RNNs have advantages in dealing with predicting nonlinear and complex patterns from historical data. However, a traditional RNN suffers from the vanishing gradient problem. So, the LSTM network is employed to build the time series classification model. The value of $\boldsymbol{q}^{j,h}$ is used as the model input each time and $\hat{\boldsymbol{p}}^k$ is the corresponding output containing the predicted probability value of a future time point t^k.

The architecture of the LSTM network is show as Fig.4, where Δt_k is the prediction period. A LSTM network has the chain like structure, but the repeating module has a different structure from RNN. The key to LSTMs is the cell state, which is shown in Fig. 5. Instead of having a single neural network layer inside each cell, there are four layers interacting in a very special way. For multivariate time series forecasting, intermediate results from the four layers at each time point are all represented by a vector-matrix form, such as $\boldsymbol{f}^j$, $\boldsymbol{I}^j$, $\boldsymbol{g}^j$, $\boldsymbol{o}^j$ and so on. The values of $\boldsymbol{f}^j$, $\boldsymbol{I}^j$, $\boldsymbol{g}^j$ and $\boldsymbol{o}^j$ denote the different network hidden units for the prediction at t^j and $\boldsymbol{c}^j$ denotes the cell state. The value of $\hat{\boldsymbol{p}}^k$ is concluded from the value of $\boldsymbol{q}^j$ and $\hat{\boldsymbol{p}}^{k-1}$, through the combination of the four layers.

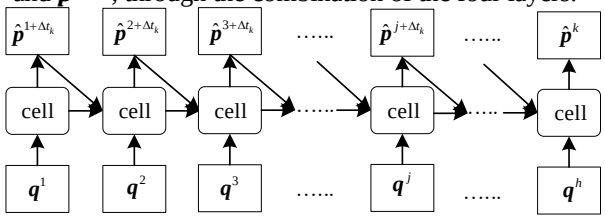

Figure 4. A LSTM network.

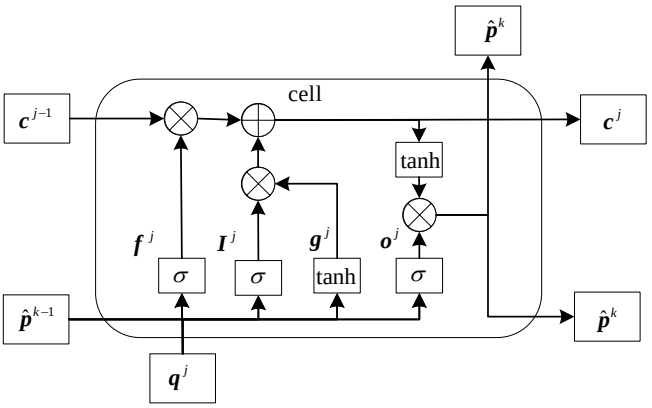

Figure 5. An example of LSTM network cell.

The multivariate LSTM model is constructed as follows. In each time point t^j, the information from the historical feature data is input along with the prediction results from the last time point t^{j-1}

$$\boldsymbol{f}^j = \sigma(\boldsymbol{W}_{qf}\boldsymbol{q}^j + \boldsymbol{W}_{pf}\hat{\boldsymbol{p}}^{k-1} + \boldsymbol{b}_f). \tag{4}$$

The next step is to decide what new information is going to store in the cell state. Firstly, a sigmoid layer, called the input gate layer, decides which values should be updated. Next, a tanh layer creates a vector of new candidate values, which could be added to the state

$$\begin{aligned} \boldsymbol{I}^j &= \sigma\left(\boldsymbol{W}_{qI}\boldsymbol{q}^j + \boldsymbol{W}_{pI}\widehat{\boldsymbol{p}}^{k-1} + \boldsymbol{b}_I\right) \\ \boldsymbol{g}^j &= tanh\left(\boldsymbol{W}_{qg}\boldsymbol{q}^j + \boldsymbol{W}_{pg}\widehat{\boldsymbol{p}}^{k-1} + \boldsymbol{b}_g\right) \end{aligned}. \quad (5)$$

In the next step, the above two states are combined to create an update to the state

$$\boldsymbol{c}^j = \boldsymbol{f}^j\boldsymbol{c}^{j-1} + \boldsymbol{I}^j\boldsymbol{g}^j. \quad (6)$$

Finally, the output is generated based on the cell state. Firstly, a sigmoid layer decides which part of the cell state is going to output. Then, the cell state is put through a tanh layer and multiplied by the output of the sigmoid gate

$$\begin{aligned} \boldsymbol{o}^j &= \sigma\left(\boldsymbol{W}_{qo}\boldsymbol{q}^j + \boldsymbol{W}_{po}\widehat{\boldsymbol{p}}^{k-1} + \boldsymbol{b}_o\right) \\ \widehat{\boldsymbol{p}}^k &= \boldsymbol{o}^j \tanh\left(\boldsymbol{c}^j\right) \end{aligned}. \quad (7)$$

For the multiple mutually-exclusive class problem, the loss L between $\widehat{\boldsymbol{p}}^k$ and $\boldsymbol{p}^k$ over all the classes is defined as the cross entropy between them

$$L = -\sum_{w=0}^{s} p_w^k \log\ \hat{p}_w^k. \quad (8)$$

The weight matrices $\boldsymbol{W}$ and the bias vectors $\boldsymbol{b}$ are adjusted through historical data training to minimize the loss L through the gradient descent. The training process continues until the loss L converges.

C. Multi-layer LSTM Classifier

While LSTM networks could handle nonlinear time series forecasting problems, a single LSTM network may perform badly on the multi-class classification problem. A multi-layer classification method is proposed in this section, following the existing work on the LSTM classifier [28], as shown in Fig. 6.

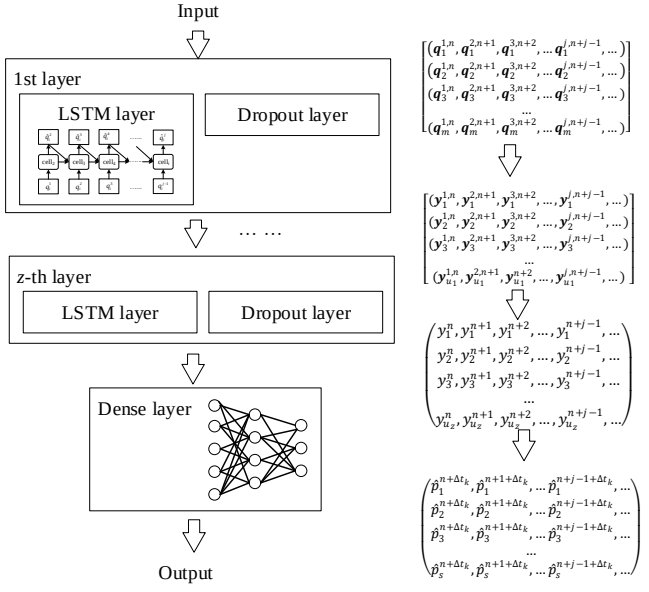

Figure 6. The aritechture of the multi-layer classification method.

The whole model is comprised of z hidden layers and a dense layer in sequence. Each hidden layer includes two steps. The first step is a specified LSTM network which accepts a 3-dimensional array and generates hidden outputs. The second step is a dropout layer which excludes some units from the LSTM randomly to avoid data over-fitting. The initial input data to the LSTM network of the first hidden layer, as well as the whole model is the 3-dimensional segmented time series sequences, shown in Fig. 3. The output of each LSTM network is another 3-dimensional array with the size of $u_i \times n \times x$, as shown in Fig. 7 (1), where $\boldsymbol{y}_i^{j,n+j-1}$ denotes the prediction result of the i-th hidden unit from time t^j to time t^{n+j-1}. Similarly, x is the number of training sequences, n is the sequence length and u_i is the number of hidden units in the i-th layer. The intermediate sequence values from the previous layer are all used as the input of the next layer, except for the LSTM in the z-th layer. The LSTM network in the last layer only generates the final output from $\boldsymbol{y}^n$ to $\boldsymbol{y}^{n+j-1}$, as shown in Fig.7 (2), where y_i^n denotes the prediction result of the i-th hidden unit at time t^n. The number of units in the z-th LSTM layer may not be the same as the number of class labels. So, a regular fully connected layer, the dense layer, is added in the end of the network. It is used as a prototype to transform the output of the last hidden layer to the vector containing probabilities of all classes, as shown by Fig.7 (3).

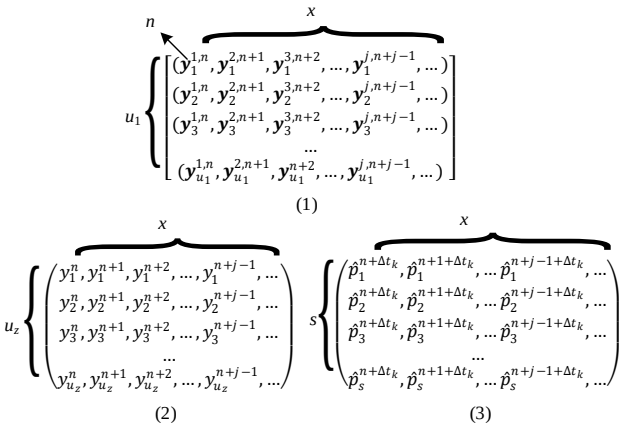

Figure 7. An example of the intemediate resluts and the final output.

V. Experimental Studies

In this section, experimental evaluation is presented for the proposed prediction method. An open-source dataset from the Microsoft Azure AI Gallery is employed for the following experiments [16]. The dataset for this experiment comes from 4 different sources. The real-time telemetry data provides the operating conditions of 100 systems from Jan 1st 2015 to Dec 31st 2015 at one-hour interval. There are over 800 thousand records in this dataset. For each system, 4 features are collected at each time interval. The system conditions, e.g. age from the first use date or history error information are also included. There are 4 components in each system which have failed during the execution and the failure history of each component is included. The previous maintenance activities of each component are also collected. The maintenance

activities include periodical preventive maintenance and the reactive maintenance upon failures. It is noted that this dataset is collected from a real-world industrial system. The collected features stand for the system execution conditions, instead of quality values. The problem of service failure prediction is similar for service-oriented systems, although the collected feature values are different.

A. Experiments Setup

There are both time-aware features such as real-time monitoring data and other features such as system age. So, data preprocessing is firstly conducted on the dataset. The time-related features are aggregated based on the feature collection time interval and the lagging time windows. Categorical feature values are reformatted into real numbers. Normalization is carried out on all the features. The prediction period Δt_k is set as 24 hours. The class labels are identified as which service is going to fail in the next 24 hours. According to the number of failed services, the number of class labels is 5. The LSTM-based classification algorithm is implemented using the Keras deep learning library. The batch size *x* and the time window *n* are selected later through experiments.

There are 4 possible outcomes for the classification results. True positive (TP) and true negative (TN) outcomes represent a correct classification, while false negative (FN) and false positive (FP) outcomes represent an incorrect one. For the algorithm evaluation, accuracy is defined as the portion of true labeled instances to the total number of instances

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (9)$$

Considering the samples imbalance among different classes, three common evaluation metrics are also used in the following experiments. Precision represents how many predicted failures are accurate for each class according to failure history. Recall presents how many failures are predicted for each class from all the failure cases. Let $Precision_w$ and $Recall_w$ denote the precision and recall values for the *w*-th class. They are defined as

$$Precision_w = \frac{TP_i}{TP_i+FP_i},\ Recall_w = \frac{TP_i}{TP_i+FN_i}, \quad (10)$$

where TP_i, TN_i, FP_i, FN_i denote the outputs for each class.

Since the precision and the recall often conflicts with each other, F1 score is used to measure the average performance between precision and recall, which is defined as:

$$F1_w = 2\frac{Pre_i * Re_i}{Pre_i+Re_i}. \quad (11)$$

Additionally, the mean values of the metrics from each class, defined as macro precision, macro recall and macro F1 score are taken use for the multi-class evaluation

$$Precision = \frac{1}{s}\sum_{w=0}^{s} Precision_w,$$
$$Recall = \frac{1}{s}\sum_{w=0}^{s} Recall_w, F1 = \frac{1}{s}\sum_{w=0}^{s} F1_w \quad (12)$$

B. Hyper-parameters Selection

For a deep learning method including the training and testing steps, the algorithm performance is highly affected by the hyper-parameters during the training process. Comparisons are conducted to choose appropriate hyper-parameters for the algorithm in this section. Those hyper-parameters having significant influences on the evaluation metrics are chosen. Due to the randomness of the deep learning methods, multiple experiments are conducted for each group of parameters to get an average metric value.

Given the prediction period Δt_k as 24 hours, the look back window or the sequence length *n* is first evaluated. Different values of sequence length and prediction period are used to train the LSTM classifier. Table 1 shows some examples of the performance metrics produced by the classifier using different values of *n* and Δt_k. It is shown in Table 1 that longer look back window and shorter prediction period will generally lead to better results. However, if the look back window is too long, e.g. twice of the prediction period, the precision and the recall values both declines. Thus, a similar value to Δt_k is the best choice for the time sequence length.

TABLE I. COMPARISONS OF MTTRICS USING DIFFERENT TIME WINDOWS

Hyper Parameter	Performance Metrics			
	Accuracy	***Macro Precision***	***Macro Recall***	***Macro F1-score***
Δt_k=24*h* *n*=12*h*	0.989470	0.931656	0.571798	0.690449
Δt_k=24*h* *n*=24*h*	0.997828	0.941206	0.942861	0.938252
Δt_k=24 *h* *n*=25*h*	**0.999092**	**0.967316**	**0.988606**	**0.977608**
Δt_k=24 *n*=48 *h*	0.998655	0.952863	0.968626	0.960081
Δt_k=12*h* *n*=24	**0.998702**	0.904160	**0.965453**	0.928418
Δt_k=24 *h* *n*=24 *h*	0.997828	**0.941206**	0.942861	**0.938252**
Δt_k=48 *h* *n*=24 *h*	0.977083	0.760975	0.712472	0.723143

After that, the number of training epochs *e* is evaluated. The training accuracy of the algorithm using different training epochs is shown in Fig. 8. It is shown that the accuracy increases dramatically with the increase of epochs in the beginning. However, the increase trend declines and the accuracy stays at a stable level after 10 epochs, even if the number of epochs continues to grow. It indicates that there is no need to choose a large number for the training epochs and the value of *e* could be no more than 10.

Apart from these parameters, the number of hidden layers *z* and the number of units in each layer u_z also play an important role in the training process. Table 2 shows some examples of the metric results using different layers and different number of units. It is shown that more layers and more units help to increase the precision and the recall values. However, it also leads to a dramatic increase in the number of network parameters and more time is needed to train the network. Taking both the training time and the algorithm

accuracy into consideration, 3 hidden layers are chosen for the LSTM network with 200 units, 100 units and 50 units each. Similarly, the dropout rate is set to be 0.2 based on the experimental results.

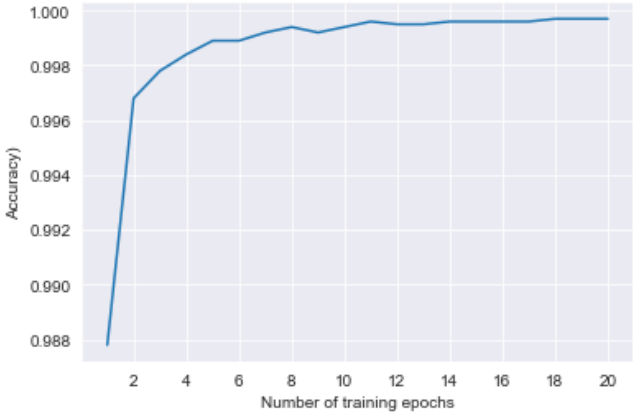

Figure 8. Comparison of accuracy using different training epochs.

TABLE II. COMPARISONS OF MTTRICS USING DIFFERENT NETWORK STRUCTURES

Hyper Parameter	Performance Metrics				No. of Parameters
	Accuracy	*Macro Precision*	*Macro Recall*	*Macro F1-score*	
u_1=50	0.998733	0.959465	0.967493	0.963059	14,055
u_1=50 u_2=25	0.998526	0.965293	0.952928	0.957214	21,530
u_1=100	0.998987	0.956719	0.983212	0.969117	48,105
u_1=100 u_2=50	0.999092	0.967316	**0.988606**	0.977608	78,055
u_1=200 u_2=100	0.999061	0.977609	0.970434	0.973460	296,105
u_1=200 u_2=100 u_3=50	**0.999307**	**0.983806**	0.980180	**0.981958**	326,055

TABLE III. EXAMPLE OF CLASSIFICATION RESULTS FOR ALL CLASSES

Performance Metrics	Class Label No.				
	None	*1*	*2*	*3*	*4*
Accuracy	0.999307	0.999307	0.999307	0.999307	0.999307
Precision	0.999626	0.941292	0.999224	0.984307	0.994580
Recall	0.999721	0.921456	1.000000	0.996470	0.983255
F1 Score	0.999674	0.931268	0.999612	0.990351	0.988885
Macro Precision	0.983806	0.983806	0.983806	0.983806	0.983806
Macro Recall	0.980180	0.980180	0.980180	0.980180	0.980180
Macro F1-score	0.981958	0.981958	0.981958	0.981958	0.981958

After the metrics evaluation, a group of hyper-parameters in the LSTM network are defined through experiments. Table III presents an example of the classification result for all 5 classes. The classification results for the class 1 is the worst in all the classes while the results for the class 0 is the best. It is because that the samples of the class 0 outnumbers the samples of all other classes in a great manner.

C. Comparison and Discussion

In this section, the performance of the LSTM-based classifier is compared with other methods. Traditionally, threshold-based methods are used to make judgements on certain time points. The range of feature values for different classes is drawn in Fig. 9. The data in Fig. 9 shows that there are no obvious differences between the features in normal state and the features in failure states at one time point. No appropriate threshold value could be chosen for each feature. The failure state is indicated by a series of feature data instead of one feature value at a time point. Thus, the traditional threshold-based method could not be used for the failure determination in this case.

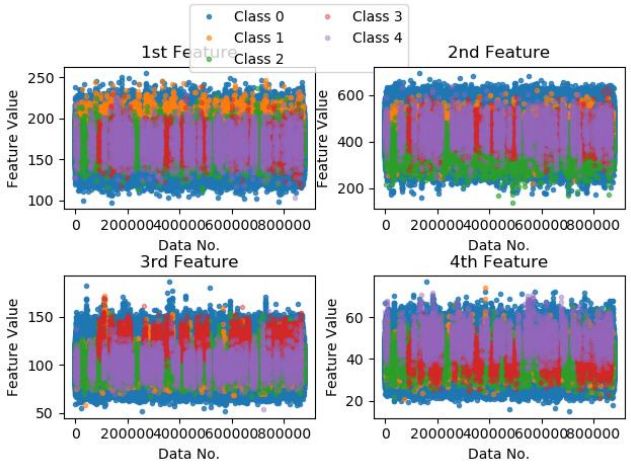

Figure 9. The rage of feature values for different classes.

In existing work, time-lagged features, instead of time series data, are used for the above problem [16]. To represent the short-term history of the features over the lag window, lag features are aggregated every 3 hours. For capturing a longer-term effect, 24-hour lag features are also calculated. Three traditional machine learning methods are used for classification, including Gradient Boosting, Random Forest and Decision Tree. To prove the algorithm efficiency, the LSTM-based classifier is compared with the above three methods in the following experiments. The whole dataset is split into the training set and the test set in 3 different ways. The results from each kind of algorithm on 3 different test sets are shown in Fig. 10. In this figure, LSTM refers to the method proposed in this paper.

Fig. 10 shows that among three kinds of machine learning algorithms, Random Forest achieves the best performance. The performance of LSTM and Random Forest is close to each other, especially in the F1 score. To make a detailed clarification, the categorical results of each metric is show in Fig. 11, taking results of test set 1 for example. The *x* axis in Fig.11 refers to the different class labels. It is shown that recall metrics of LSTM is generally better than the Random Forest

method while Random Forest performs better in the precision for some classes. Although the performance of these two algorithms is close, the LSTM network is more practical. It is built on the original time series instead of engineered features. On the contrary, much more efforts are made on selecting and transforming time series features before conducting Random Forest classification since it is difficult to describe system condition exactly using discrete features.

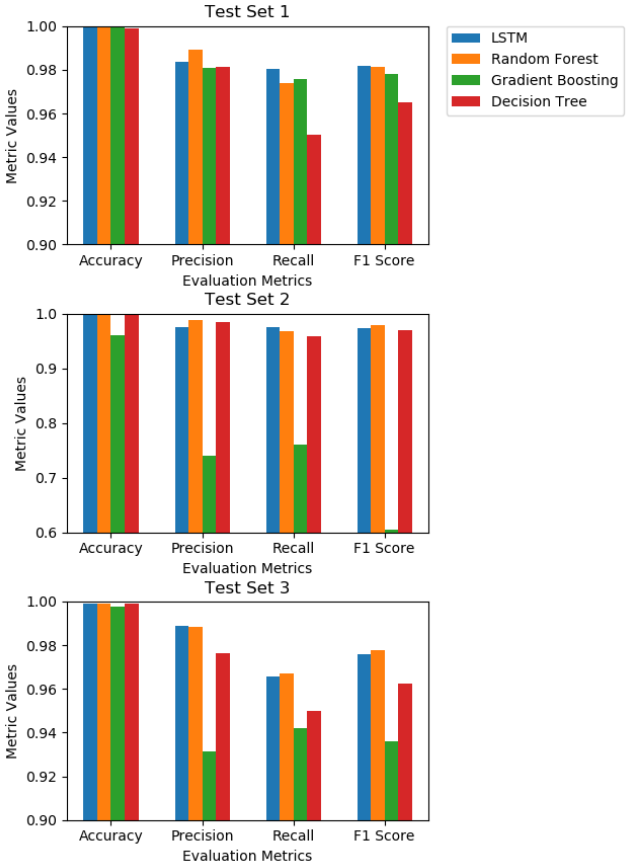

Figure 10. Comparisons of different classificatin algorithms.

VI. Conclusions and Future Work

To reduce the costly impact caused by downtime in IoT systems, proactive or predictive approaches are used as the primary approach. It prevents the occurrence of failure by predicting time to failure and estimating the optimum time to do maintenance. Thus, online failure prediction plays an important role in the whole process. To predict whether a component is about to fail in the certain amount of time, this paper proposes a multi-layer data-driven failure prediction model for proactive maintenance. The service failure prediction problem is cast into a time-lagged multi-class classification problem. The time-series quality data, historical error and maintenance data are all used as the model features while the time-lagged service failure information is employed as the class label. A multi-layer LSTM based approach is used to solve the time-lagged multi-class classification problem. The prediction result indicates the specific service to fail as well as the fact that whether the service is going to fail in the remaining time. The prediction method is experimented on an open IoT maintenance dataset and compared with some existing machine learning methods. The comparison results show that without the additional feature engineering work, the LSTM based approach outperforms most of the traditional machine learning methods in both precision and recall metrics. In the next step of work, the algorithm performance should be further improved without affecting the overall accuracy. For example, the structure of LSTM network could be redesigned to decrease the number of layers and units.

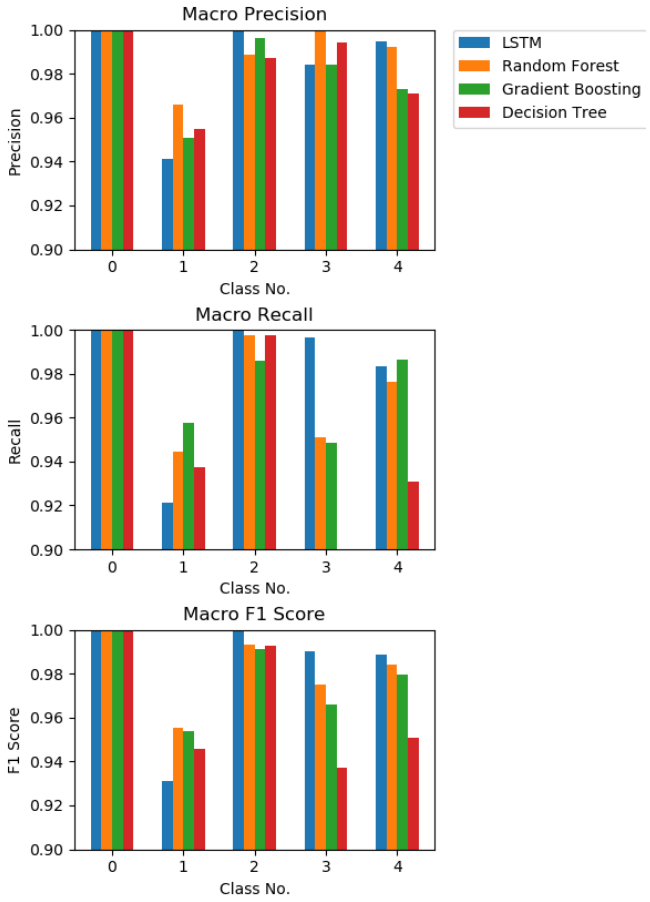

Figure 11. Comparisons of different classificatin algorithms usig one test set.

Acknowledgment

This research was financially supported by the Basic Research and Frontier Exploration (Natural Science Foundation) Projects in Chongqing (Grant no. cstc2019jcyj-msxmX0442), the Key Technology Innovation and Application Demonstration Projects in Chongqing (Grant no. cstc2018jszx-cyzdX0068), and the Technology Innovation and Application Development Project in Chongqing (Grant no. cstc2019jscx-msxmX0051).

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *Ieee Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015 2015, doi: 10.1109/comst.2015.2444095.

[2] N. Joneshttps, "Top Strategic IoT Trends and Technologies Through 2023," 2018/09/21 2018. [Online]. Available: https://www.gartner.com/en/documents/3890506/top-strategic-iot-trends-and-technologies-through-2023.

[3] F. Civerchia, S. Bocchino, C. Salvadori, E. Rossi, L. Maggiani, and M. Petracca, "Industrial Internet of Things monitoring solution for advanced predictive maintenance applications," (in English), *J. Ind. Inf. Integr.*, Article vol. 7, pp. 4-12, Sep 2017, doi: 10.1016/j.jii.2017.02.003.

[4] R. R. Aschoff and A. Zisman, "Proactive adaptation of service composition," presented at the Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Zurich, Switzerland, 2012.

[5] N. K. Kahlon, S. V. Kapur, K. K. Chahal, and S. B. Narang, "A Proactive Solution to Manage Web Service Unavailability in Service Oriented Software Systems," Berlin, Heidelberg, 2016: Springer Berlin Heidelberg, in Service-Oriented Computing – ICSOC 2015 Workshops, pp. 243-254.

[6] H. Wang, L. Wang, Q. Yu, Z. Zheng, and Z. Yang, "A proactive approach based on online reliability prediction for adaptation of service-oriented systems," *Journal of Parallel and Distributed Computing*, vol. 114, pp. 70-84, Apr 2018, doi: 10.1016/j.jpdc.2017.12.006.

[7] F. Salfner, M. Lenk, and M. Malek, "A Survey of Online Failure Prediction Methods," *Acm Computing Surveys*, vol. 42, no. 3, Mar 2010, Art no. 10, doi: 10.1145/1670679.1670680.

[8] S. T. March and G. D. Scudder, "Predictive maintenance: strategic use of IT in manufacturing organizations," *Information Systems Frontiers*, journal article vol. 21, no. 2, pp. 327-341, April 01 2019, doi: 10.1007/s10796-017-9749-z.

[9] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Comput. Ind. Eng.*, vol. 137, p. 106024, 2019/11/01/ 2019, doi: https://doi.org/10.1016/j.cie.2019.106024.

[10] H. Jingjing, C. Xiaolei, and Z. Changyou, "Proactive service selection based on acquaintance model and LS-SVM," *Neurocomputing*, vol. 211, pp. 60-65, 2016/10/26/ 2016, doi: https://doi.org/10.1016/j.neucom.2015.11.119.

[11] W. Zhang, D. Yang, and H. Wang, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213-2227, 2019, doi: 10.1109/JSYST.2019.2905565.

[12] Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-Term QoS-Aware Cloud Service Composition Using Multivariate Time Series Analysis," *Ieee Transactions on Services Computing*, vol. 9, no. 3, pp. 382-393, May-Jun 2016, doi: 10.1109/tsc.2014.2373366.

[13] H. Wang, Z. Yang, Q. Yu, T. Hong, and X. Lin, "Online reliability time series prediction via convolutional neural network and long short term memory for service-oriented systems," *Knowledge-Based Systems*, vol. 159, pp. 132-147, 2018/11/01/ 2018, doi: https://doi.org/10.1016/j.knosys.2018.07.006.

[14] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, and M. R. Lyu, "A Spatial-Temporal QoS Prediction Approach for Time-aware Web Service Recommendation," *ACM Trans. Web*, vol. 10, no. 1, p. Article 7, 2016, doi: 10.1145/2801164.

[15] Y. Syu, C.-M. Wang, and Y.-Y. Fanjiang, "A Survey of Time-Aware Dynamic QoS Forecasting Research, Its Future Challenges and Research Directions," Springer International Publishing, 2018, pp. 36-50.

[16] F. B. Uz, "Predictive Maintenance Modelling Guide Data Sets," 2016/03/29 2016. [Online]. Available: https://gallery.azure.ai/Experiment/Predictive-Maintenance-Modelling-Guide-Data-Sets-1.

[17] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," (in English), *Automat. Softw. Eng.*, Article vol. 15, no. 3-4, pp. 313-341, Dec 2008, doi: 10.1007/s10515-008-0032-x.

[18] N. K. Kahlon, K. K. Chahal, and S. B. Narang, "Managing QoS Degradation of Component Web Services in a Dynamic Environment," (in English), *Int. J. Semant. Web Inf. Syst.*, Article vol. 14, no. 2, pp. 162-190, Apr-Jun 2018, doi: 10.4018/ijswis.2018040108.

[19] Y. Syu, C. M. Wang, and Y. Y. Fanjiang, "A Survey of Time-Aware Dynamic QoS Forecasting Research, Its Future Challenges and Research Directions," in *Services Computing - Scc 2018*, vol. 10969, J. E. Ferreira, G. Spanoudakis, Y. Ma, and L. J. Zhang Eds., (Lecture Notes in Computer Science. Cham: Springer International Publishing Ag, 2018, pp. 36-50.

[20] Y. Dai, L. Yang, and B. Zhang, "QoS-Driven Self-Healing Web Service Composition Based on Performance Prediction," *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 250-261, 2009/03/01 2009, doi: 10.1007/s11390-009-9221-8.

[21] H. Wang, L. Wang, Q. Yu, Z. Zheng, A. Bouguettaya, and M. R. Lyu, "Online Reliability Prediction via Motifs-Based Dynamic Bayesian Networks for Service-Oriented Systems," *Ieee Transactions on Software Engineering*, vol. 43, no. 6, pp. 556-579, Jun 1 2017, doi: 10.1109/tse.2016.2615615.

[22] R. B. Xiong, J. Wang, Z. Q. Li, B. Li, P. C. K. Hung, and Ieee, *Personalized LSTM Based Matrix Factorization for Online QoS Prediction* (2018 Ieee International Conference on Web Services). New York: Ieee (in English), 2018, pp. 34-41.

[23] R. Aschoff and A. Zisman, "QoS-Driven Proactive Adaptation of Service Composition," in *Service-Oriented Computing*, vol. 7084, G. Kappel, Z. Maamar, and H. R. MotahariNezhad Eds., (Lecture Notes in Computer Science, 2011, pp. 421-435.

[24] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, Prediction and Prevention of SLA Violations in Composite Services," in *2010 IEEE International Conference on Web Services*, 5-10 July 2010 2010, pp. 369-376, doi: 10.1109/ICWS.2010.21.

[25] E. Mosayebi Omshi, A. Grall, and S. Shemehsavar, "A dynamic auto-adaptive predictive maintenance policy for degradation with unknown parameters," *European Journal of Operational Research*, vol. 282, no. 1, pp. 81-92, 2020, doi: 10.1016/j.ejor.2019.08.050.

[26] J.-R. Ruiz-Sarmiento, J. Monroy, F.-A. Moreno, C. Galindo, J.-M. Bonelo, and J. Gonzalez-Jimenez, "A predictive model for the maintenance of industrial machinery in the context of industry 4.0," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103289, 2020/01/01/ 2020, doi: https://doi.org/10.1016/j.engappai.2019.103289.

[27] S. Zhang, C. Liu, S. Su, Y. Han, and X. Li, "A feature extraction method for predictive maintenance with time-lagged correlation-based curve-registration model," *International Journal of Network Management*, vol. 28, no. 5, Sep-Oct 2018, Art no. e2025, doi: 10.1002/nem.2025.

[28] K. T. P. Nguyen and K. Medjaher, "A new dynamic predictive maintenance framework using deep learning for failure prognostics," *Reliability Engineering & System Safety*, vol. 188, pp. 251-262, 2019/08/01/ 2019, doi: https://doi.org/10.1016/j.ress.2019.03.018.

[29] L. Guo, P. Wan, R. Li, G. Liu, and P. He, "Runtime Quality Prediction for Web Services via Multivariate Long Short-Term Memory," (in English), *Math. Probl. Eng.*, Article vol. 2019, Aug 2019, Art no. 2153027, doi: 10.1155/2019/2153027.