An Anomaly Detection Method for Satellites Using Monte Carlo Dropout

Recently, there has been a significant amount of interest in satellite telemetry anomaly detection (AD) using neural networks (NN). For AD purposes, the current approaches focus on either forecasting or reconstruction of the time series, and they cannot measure the level of reliability or the probability of correct detection. Although the Bayesian neural network (BNN)-based approaches are well known for time series uncertainty estimation, they are computationally intractable. In this article, we present a tractable approximation for BNN based on the Monte Carlo (MC) dropout method for capturing the uncertainty in the satellite telemetry time series, without sacrificing accuracy. For time series forecasting, we employ an NN, which consists of several long short-term memory (LSTM) layers followed by various dense layers. We employ the MC dropout inside each LSTM layer and before the dense layers for uncertainty estimation. With the proposed uncertainty region and by utilizing a postprocessing filter, we can effectively capture the anomaly points. Numerical results show that our proposed time series AD approach outperforms the existing methods from both prediction accuracy and AD perspectives.

I. INTRODUCTION

Satellites are complicated systems composed of various interconnected technologies such as telemetry sensing, mobile communications, and navigation systems [1]. Proactive diagnosis of failures, anomaly detection (AD), and response to potential hazards are required to guarantee the availability and continuity of satellite services [2]. Considering the complex design structure and harsh environment of the space, AD on the satellites cannot be performed directly in the outer space; instead, the telemetry data are employed to provide healthy key information that can be used to recover the satellite from the possible problems [3]. Previous satellite AD systems were based on expert knowledge. However, by receiving a large amount of data from different telemetry channels, the expert-based (human-based) AD
systems cannot properly discover anomalies, so data-driven intelligent AD methods are recommended [4].

Many studies have focused on deep neural networks (DNN) for data-based AD for spacecraft applications [5], [6]. AD in this context has been studied from two different standpoints, which are as follows:

1) **Forecasting-based method:** In this approach, the prediction error is computed as a difference between the current and the predicted states. Then, by leveraging a thresholding-based method, an anomaly can be detected. The idea behind this approach is that an efficient detection method should learn the expected behavior of a telemetry channel, so any deviation from the expected response can be flagged as a tentative anomaly [7], [8], [9], [10], [11], [12], [13], [14], [15].

2) **Reconstruction-based method:** In this AD approach, time series data are reconstructed, and then by comparing the real (true) and reconstructed values, anomalies are detected [15], [16], [17], [18]. As this approach must reconstruct the time series to detect anomalies, it cannot be performed in real time.

Although both reconstruction and forecasting-based approaches can capture the anomaly points, they cannot provide confidence intervals of an NN model. However, model uncertainty is essential for assessing how much to rely on the forecast produced by the model and it plays a critical role in applications like AD. The uncertainty measure can be the variance or confidence intervals around the prediction made by the NN. In this article, we use MC dropout to capture the variance and to construct confidence interval for the NASA satellite telemetry dataset [7]. Specifically, by using MC dropout, we derive the first- and second-order statistics of an NN consisting of the LSTM cells and the dense layers. These statistics are used to determine which points fall inside the confidence region and flag tentative points that are outside the region. In this article, we describe how to use Bayesian LSTM for fault detection, identification, and recovery (FDIR) for spacecrafts. We show the proposed MC-dropout approach outperforms the existing methods [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] and it has several advantages, which will be discussed in detail in the following sections.

The rest of this article is organized as follows: in Section II, we present the related works and lay out our contributions. In Section III, we discuss the proposed preprocessing method. Bayesian LSTM methods for training and postprocessing AD filters are presented in Section IV. Performance evaluation of the proposed AD method is presented in Section V finally, Section VI concludes this article.

II. RELATED WORKS

In this section, we first review the literature on different AD methods, compare the existing works from various perspectives, and then discuss this article’s contributions.

A. Literature Review

While forecasting-based methods suffer from sensitivity to parameter selection and often require strong assumptions and extensive domain knowledge about the data, they grant a significant amount of interest due to their low complexity. In this respect, some statistical-based models have been proposed, such as autoregressive integrated moving average (ARIMA), which is an analytical model that learns the time series autocorrelation for future value predictions [10]. Furthermore, different sequential based deep neural network (NN) approaches such as LSTM, DeepAR, HTM are also forecasting-based methods [12], [13], [14]. In [13], the authors introduce hierarchical temporal memory (HTM) for AD in streaming data. The HTM encodes the current input to a hidden state and predicts the next hidden state. In [12], the performance of LSTM for detecting anomalies in the space shuttle dataset is discussed. In [14], the authors used DeepAR, which is based on autoregressive recurrent networks. In this approach, anomalies are computed as the regression errors, which are the distance between the median of the predicted and the true values.

On the other hand, reconstruction-based methods aim to learn a model in order to capture the low-dimensional structure of the data. Different from the normal points, the anomalies lose more information when mapping to a lower dimensional space, so, as a result, they cannot be effectively reconstructed. Thus, the points with high reconstruction errors are more likely to be anomalies. In this respect, generative adversarial networks (GANs) for time series AD are proposed in [16] and [18]. The OmniAnomaly method in [15] proposes a stochastic recurrent neural network (RNN), which captures the normal patterns of multivariate time series by modeling data distribution through stochastic latent variables. Furthermore, in [7], the LSTM Auto-Encoder (AE) efficiency for detecting the satellite anomalies is demonstrated. Recently, in [19], the authors proposed a reconstruction-based AD method based on a combination of AE and convolutional NN to capture the temporal correlations and spatial features of the multivariate time series. In [20], an implicit neural representation-based AD approach is utilized. In [21] and [22], a deep transformer NN which uses attention-based sequence encoders for AD is developed. More specifically, in [22] the prediction accuracy and AD are shown to be significantly improved by using an attention-based mechanism.

Nevertheless, both reconstruction and forecasting-based approaches are shown promising results in AD, they cannot provide a confidence region of the predicted points by NN. In the context of deep NN, the uncertainty region is defined as the likelihood interval for the true value that the NN prediction lies within, and the main approach to construct this interval is Bayesian Neural Network (BNN) [23]. Using this uncertainty region and predicted value by NN, we can determine the anomaly points. As BNN does not suffer from over-fitting and always offers uncertainty estimation, it grants a significant amount of interest in many applications [24]. However, finding the posterior distribution of
BNN is challenging and computationally intractable. Therefore, the posterior distribution needs to be approximated with different techniques [24], [25], [26], [27], [28]. Among various approximation approaches, the MC dropout has demonstrated important benefits, such as lower computational cost and higher precision, over other methods [24]. The usage of MC dropout for RNN (i.e., LSTM) and the mathematical formulation can be found in [29]. In [1], the authors proposed an AD technique based on an approximation of BNN. Furthermore, in [30] and [31], application of BNN for AD is discussed. Moreover, in [32], the authors used the inference abilities and modeling characteristics of dynamic Bayesian networks in developing and implementing an innovative approach for FDIR of autonomous spacecraft.

B. Contributions

In this section, we discuss the contributions, advantages, and major differences of the proposed Bayesian method compared to other competitors. The key contributions of this article are summarized as follows:

1) We propose an unsupervised Bayesian AD method for time series data. In particular, we use the MC dropout as a low-complexity approximation of BNN in both the dense and LSTM layers of the NN for AD tasks. The proposed Bayesian method not only forecasts the data accurately but also provides a confidence interval as a byproduct. Note that, our approximation of BNN is more precise than the method in [1]. The MCD-BiLSTM-VAE method in [1] uses dropout only before the dense layers rather than the entire network composed of both the LSTM cells and the dense layers, and thus it only captures the uncertainty of the dense layers. In contrast, we applied MC dropout at both of the LSTM and dense layers. Indeed, to get a precise approximation for BNN, the uncertainty of both LSTM and the dense layers should be acquired simultaneously [24], [29]. Furthermore, another significant difference between our approach and [1] in AD is that the thresholding approach in [1] is static, whereas we propose a dynamic thresholding method. For the dataset under consideration, the dynamic thresholding method surpasses the approach of [1] in terms of more accurate prediction and better classification results.

2) We propose a preprocessing method, which is based on an adaptive weighted average inside a time window with a variable length to smooth the time series. In our preprocessing method, unlike the standard preprocessing steps such as normalization and data cleaning, which are implemented in [1], [7], [11], [16], and [17], we first clean, normalize, and scale the dataset, then implement an innovative algorithm. To the best of our knowledge, no prior work can be found that uses this method.

3) The proposed Bayesian method is a low complexity approach in terms of training time and throughput. In fact, mathematically, it can be shown that the dropout itself reduces the complexity in terms of the number of operations during both of the training and testing processes [33]. Moreover, comparing to the reconstruction-based methods such as [1], [7], [16], our method is less computationally complex since it uses a smaller portion of the time series for predicting the next step. Indeed, at each time step, we use a number of data points equal to the order of the Markov model (which in practice will generally be one or two points) to estimate the next step and detect tentative anomalies. In contrast, reconstruction-based methods utilize all the time-series samples for AD. Consequently, they cannot be implemented in the real time.

4) Our proposed method prevents the model from overfitting, which is due to adding the regularization term to the loss function [33]. This is different from the methods in [7], [11], [16], and [17], which might suffer from overfitting.

5) We propose a postprocessing algorithm, which is based on the divergence of a succession of predicted data points from the corresponding confidence intervals. This is different from the approaches in [1], [7], and [16], which suffer from a high number of false positives. The proposed postprocessing approach is shown to lessen the number of false positives and, as a result, improves the accuracy of AD.

III. PREPROCESSING FOR ANOMALY DETECTION

The raw telemetry data could not be directly ingested into our AD algorithm, and it should first go through a preprocessing stage [7]. An special weighted moving average (WMA) filter is adopted to improve the prediction accuracy by making the time series smoother [34]. In our terminology, the high prediction accuracy is equivalent to low mean square error (MSE) between the real values and the predicted values. However, when there exist anomaly points inside a window of the time series, the standard WMA may not be applicable. In fact, due to the averaging operation between the current and the adjacent points, there exists a high chance of missing an anomaly point. Our motivation here is to devise a strategy to take the advantages of the WMA for improving the prediction accuracy without missing any anomaly point. In fact, we average among the consecutive data points within an adaptive rolling window with a variable length. The idea lies in the fact that when the difference between two consecutive measurements is high, there exists a high chance of having an outlier/anomaly point. Therefore, a narrower window length should be used, as we do not want the anomaly point to be missed before applying the anomaly detection algorithm. Alternatively, when the difference between two consecutive points is small, we choose a wider window to make the time series smoother. Assume that \( \mathbf{u} \in \mathbb{R}^{nx1} \) and \( \mathbf{x} \in \mathbb{R}^{nx1} \) are the raw data and the preprocessed data, respectively. Let \( l_w(k) \) be the length of the \( k \)th window of our filter. Assume
that the processed data in the kth window is x(k), where k ∈ J, J = {1, . . . , n}. The smoothed data is given by

\[
x(k) = \frac{1}{\|k\|} \sum_{i=-\frac{\|k\|}{2}}^{\frac{\|k\|}{2}} \alpha_i u(k-i)
\]  

(1)

where \( \kappa = [a_{\omega_u}, \ldots, a_{\omega_u}]^T \) is the vector norm function, and \( \alpha_i, i \in \mathcal{I} = \{-\frac{|\kappa|}{2}, \ldots, \frac{|\kappa|}{2}\} \) are the smoothed filter coefficients. In our approach, the smoothing coefficients \( 0 \leq \alpha_i \leq 1 \) depend on the distance function \( d \), where \( d(k-i) = \|u(k-i-1) - u(k-i)\| \). The distance function is the absolute difference between two successive data points inside the kth window. In this preprocessing algorithm, we start with \( l_w(0) = 2 \) and \( \alpha_0 = 1 \). Let the distance threshold value be \( d_{th}(k) = m(k) + 2\sigma(k) \), where \( m(k) \) and \( \sigma(k) \) are the mean and standard deviation (s.t.d.) of the data inside the kth window. We choose the threshold value \( d_{th}(k) = m(k) + 2\sigma(k) \) to take into account both the mean and variance inside each window. However, this threshold can be further improved by examining the exact distribution of different time series and choosing the optimal value for the threshold. It remains an interesting avenue for our future studies.

For the ith data in the kth window, if \( d(k-i) < d_{th}(k) \) (i.e., we do not see a sudden change), we increase the window length and set \( \alpha_i = 1, \ i \in \mathcal{I} \). Otherwise, if \( d(k-i) \geq d_{th}(k) \), we have

\[
\alpha_i = \begin{cases} 
0, & i \in \mathcal{I} - \{0\} \\
1, & i = 0, 
\end{cases}
\]  

(2)

then we increment k, and proceed to another window, i.e., \( k = k + 1 \). The preprocessing procedure is shown in Algorithm 1.

**Algorithm 1: Preprocessing Algorithm.**

**Input:** u(k).

**Output:** x(k).

1: Initialization: \( l_w(0) = 2, \alpha_0 = 1, k = 0 \).
2: while \( k \leq n \) do
3: Calculate the mean \( m(k) \) and s.t.d. \( \sigma(k) \) of the values inside the kth windows with length \( l_w(k) \).
4: Select the kth threshold as \( d_{th}(k) = m(k) + 2\sigma(k) \).
5: for \( i \in \mathcal{I} \) do
6: if \( d(k-i) < d_{th}(k) \) then
7: \( \alpha_i = 1 \)
8: \( l_w(k) = l_w(k) + 1 \)
9: else if \( d(k-i) > d_{th}(k) \) then
10: Break
11: end if
12: end for
13: Calculate \( x(k) = \frac{1}{\|k\|} \sum_{i=-\frac{\|k\|}{2}}^{\frac{\|k\|}{2}} \alpha_i u(k-i) \).
14: Go to the next window, i.e., \( k = k + 1 \).
15: end while

**IV. BAYESIAN LSTM METHOD FOR PREDICTION AND UNCERTAINTY ESTIMATION**

Here, we discuss a Bayesian LSTM approach for predicting the mean of the model and its associated variance (uncertainty level). Fig. 1 shows our proposed NN model and the underlying Bayesian approach, where we use several LSTM cells followed by the dense layers. In this section, we adopt the MC dropout for approximation of the BNN model in both of the LSTM and dense layers separately.

**A. MC Dropout as an Approximation of BNN**

We assume an NN with L LSTM and D dense layers. A batch of T observations is passed through the input layer. By assuming \( T_D \) as the length of the Dth dense layer, we denote \( x \in \mathbb{R}^T \) and \( y \in \mathbb{R}^{T_D} \) as the input and output of NN, respectively. Assume that \( \omega \) with prior distribution \( p(\omega) \) represents a collection of the NN parameters. The probability of the output vector given the data for an associated input vector \( x \) is

\[
p(y \mid x, D) = E_{p(\omega \mid D)} \left( p(y \mid x, \omega) \right)
\]  

(3)

where \( D = \{(x_t, y_t) \ \forall t \in \{1, \ldots, T\}\} \) and \( p(\omega \mid D) \) are a batch of data, and the posterior distribution on weights, respectively. The expectation under \( p(\omega \mid D) \) is equivalent to using an ensemble of an infinite number of models, which is indeed computationally intractable.

One feasible solution is to approximate the posterior distribution over the model parameters \( p(\omega \mid D) \) with a simpler distribution \( q(\omega) \). A metric of similarity (or distance) between two probability distribution functions (PDF) is Kullback–Leibler (KL) divergence. Here, we minimize the KL distance between \( p(\omega \mid D) \) and \( q(\omega) \) [24]. More specifically, we solve the following optimization problem:

\[
\min_{q(\omega)} \text{KL}(q(\omega) \| p(\omega \mid D))
\]  

(4)
which is equal to minimizing the following:

$$\min_{q(\omega)} - \int q(\omega) \log p(y| x, \omega) d\omega + KL(q(\omega)\|p(\omega)).$$  \hspace{1cm} (5)

where $p(\omega)$ stands for the prior distribution of the NN parameters. As seen in (5), the objective function is composed of two parts. The first component is the NN loss function and the second one stands for the regularization effect, which prevents the model from being overfitted. Minimizing the second component is equivalent to finding a $q(\omega)$ close to the prior, which essentially avoids overfitting. The first component, on the other hand, can be rewritten with the MC sampling over $\omega$ with a single sample as the following:

$$\frac{1}{T} \sum_{t=1}^{T} \log p(y_t| f^w(x_t)) d\omega. \hspace{1cm} (6)$$

In the next section, we discuss the detailed structure of the function $f^w(x_t)$ in (6) using different layers of LSTM followed by dense layers.

B. Application of MC Dropout for Approximation of the Posterior Distribution of NN Parameters

As shown in Fig. 1, a simple LSTM unit contains input, output, and additional control gates. The following formulation briefly shows the principle of the LSTM [35]:

$$i_t = \sigma(W_i x_t + b_i), \quad f_t = \sigma(W_f x_t + b_f), \quad o_t = \sigma(W_o x_t + b_o), \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t) \hspace{1cm} (7)$$

where $x_t = [x_t, h_{t-1}]^T$ and $x_t$ represents the input data, $\sigma$ is the logistic sigmoid function, and $\odot$ is the elementwise product. Also, $c_t$ represents the state value of the LSTM unit (while $\tilde{c}_t$ is the candidate state value), $i_t$ stands for the state value of the input gate, $f_t$ represents the state value of the forget gate, $o_t$ is the state value of the output gate, $h_t$ is the output of the LSTM unit, and $t$ is the current time step. $W$ and $b$ are the corresponding weight and bias parameters at each of the aforementioned gates in (6). According to the aforementioned formulation of LSTM, the following mapping function can be concluded (omitting the bias parameter):

$$h_t = f_h(x_t). \hspace{1cm} (8)$$

By applying a dropout at the input and hidden layers, we have [29]

$$i_t = \sigma(W_i x_t + b_i), \quad f_t = \sigma(W_f x_t + b_f), \quad o_t = \sigma(W_o x_t + b_o), \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t) \hspace{1cm} (9)$$

where $m \sim B(p)$ is a random mask vector where each row has a Bernoulli distribution with the parameter $p$ and it is repeated at all the time steps. We use the MC dropout in the dense layers by dropping the neurons randomly according to the Bernoulli distribution during the testing phase. For the $k$th dense layer, we have [24]

$$W_k = \text{diag} \left( \left[ \frac{z_{k,j}}{K_k} \right]_{j=1}^{K_k} \right) M_k, \quad z_{k,j} \sim B(p_k), \quad \forall k \in \{1, \ldots, K_k\}, \quad j \in \{1, \ldots, K_k-1\} \hspace{1cm} (10)$$

where $p_k$ and matrix $M_k$ of dimensions $K_k \times K_k-1$ are the variational parameters. The binary variable $z_{k,j} = 0$ corresponds to the unit $j$ in layer $k-1$ being dropped out as an input to the $k$ layer. By assuming $b_i$ as a bias at the $k$th dense layer, the weight matrix (NN parameter) can be considered as $\omega = [W_k, W_t, W_f, W_o, W_\tilde{c}, b_k, b_i, b_f, b_o, b_c]^T$. Therefore, (6) is rewritten as

$$\sum_{i=1}^{N} \log \left( p\left( y_i \left| \sqrt{\frac{1}{K_D}} \tilde{W}_D \sigma \left( \left[ \begin{array}{c} \cdots \frac{1}{\sqrt{K_2}} \tilde{W}_2 \sigma (W_T (\psi_T) + b_1) \end{array} \right] \right) \right) \right) \hspace{1cm} (11)$$

where $\psi_T \stackrel{\Delta}{=} [x_{T,i}, f_{h}^{\omega} (\cdots f_{h}^{\omega} (x_{T,i}, b_0))]$, and $\omega \sim q(\omega)$. For each sequence $x_t$, we sample a new realization $\tilde{\omega} = [\tilde{W}_k, \tilde{W}_t, \tilde{W}_f, \tilde{W}_o, \tilde{W}_c, \tilde{b}_k, \tilde{b}_i, \tilde{b}_f, \tilde{b}_o, \tilde{b}_c]^T$. Each symbol in the sequence $x_t = [x_{t,1}, \ldots, x_{t,T}]^T$ is passed through the function $f_{h}^{\omega}$ with the same weight realizations $\tilde{\omega}$ used at the time step $t \leq T$. In the dense layers, and LSTM layers, the MC dropout is performed during both of the training and testing phases. To summarize, in MC dropout, we set the input of each neuron (or LSTM cell) independently to zero with probability $p$, which means running the network several times with different random seeds. Algorithm 2 shows the different steps of the MC dropout approach.

Assume that we have $l$ sets of realization of the NN after applying the MC dropout, as shown in Fig. 1. As derived in [24], the approximated predictive distribution is given by the following:

$$q(y|x) = \int p(y|x, \omega) q(\omega) d\omega. \hspace{1cm} (12)$$

The first two moments are derived as [24]

$$E_{q(y|x)}(y) \approx \frac{1}{l} \sum_{i=1}^{l} E_{q(\omega)}(y|x, \omega^i) \hspace{1cm} (13)$$
and
\[ E_{q(y|x)}(y^T y) \approx \frac{1}{l} \sum_{t=1}^{l} E(y|x, \omega')^T E(y|x, \omega'). \]  
(18)

The model’s predictive variance is
\[ \text{Var}_{q(y|x)}(y) = E_{q(y|x)}(y^T y) - E_{q(y|x)}(y)^T E_{q(y|x)}(y). \]  
(19)

C. Postprocessing of AD

To reduce the number of false positives, we propose a postprocessing algorithm, which is based on the deviation of a sequence of predicted data points from the confidence region. More specifically, if a predicted data point is outside the uncertainty region of the MC dropout, then it is considered as a tentative anomaly point. In our specific dataset, anomaly points do not occur as a single point; instead, they emerge as a sequence (burst) of dependent points in a particular part of the time series.

It is important to determine a time when the anomaly starts. Let \( N_{\text{max}} \) be a range of consecutive data points in time series. In our approach, if \( 0.8N_{\text{max}} \) of the consecutive points are outside the confidence region simultaneously, then we consider the first index of the sequence as the starting point of the anomaly sequence. For this specific dataset, this postprocessing approach can significantly reduce the number of false positives. Choosing a suitable value for \( N_{\text{max}} \) depends on the maximum allowable delay. We can improve the accuracy if we wait for more sample points to announce the anomaly; however, successful alert with a huge delay is not useful. Therefore, if an AD algorithm raises an alert fast enough (i.e., before a maximum allowable delay) the whole anomaly fragment is considered to be detected successfully. We will further illustrate this method in numerical results.

V. NUMERICAL RESULTS

First, we describe the methodology for evaluating AD, then summarize the specific dataset under consideration, followed by the performance evaluation. The source code is available at.\(^1\)

A. Methodology

We assume an NN with 3 LSTM and 2 dense layers, where the MC dropout rate is assumed to be 0.2. We adopt PyTorch, an open-source framework for machine learning, to implement our proposed approach. We use the dataset in [7] to compare and evaluate our method with the following approaches: i) dynamic thresholding approach with LSTM Auto-Encoder (AE) [7]; ii) TadGan approach [16]; iii) MadGan method [17]; iv) Arima method [10]; v) LSTM method [11]; and vi) A modified version of MCD-BiLSTM-VAE method in [1] after applying the postprocessing filter, which has been proposed in Section IV-C. We will use the following metrics to evaluate the performance of various models: mean squared error (MSE), \( F_1 \) score, recall, and precision. Moreover, we will compare the computational complexity of different methods. These metrics have been widely used in the literature and are discussed in detail in [36].

B. NASA Dataset

We use the satellite telemetry dataset, which was originally collected by NASA [7]. The dataset comes from two spacecrafts: the Soil Moisture Active Passive (SMAP) and the Curiosity Rover on Mars (MSL). There are 82 signals available in the NASA dataset, which are summarized in Table I. We found 54 of the 82 signals (multivariate time series) are continuous, and the remaining are discrete. There are a total of \( n \) rows corresponding to the number of timestamps or readings taken for that signal (\( n \) differs in different signals). Each sample of the data consists of one column of telemetry signal and several columns of commands. The command features are encrypted and can barely be used. Furthermore, we also found that the correlation between the command features and telemetry signals is negligible. We consider only the time series sequences from the telemetry signals in our evaluation. Therefore, the input signal to our model is an \( n \times 1 \) matrix.

C. Evaluation

In this section, we evaluate and compare the proposed model with those of other studies. The comparison results between the predicted and real values in the training and testing phases for one of the MSL satellite telemetry signals, i.e., “F-7,” and another SMAP satellite signal, i.e., “P-1” are shown in Figs. 2 and 3, respectively. By analyzing these two figures, we notice that our Bayesian LSTM approach performs better on the SMAP data in the prediction phase. To show the uncertainty region by our Bayesian approach, Figs. 4 and 5 are depicted. These figures show the MC dropout bounds and corresponding predicted values for “F-7” from MSL dataset and “P-1” from SMAP dataset, respectively. We found that, on average, for both datasets, 84%

\(^1\)[Online]. Available: https://bit.ly/3zf11Pb
Fig. 3. Examples of real values and predictions in training and testing modes for “P-1” signal.

Fig. 4. Uncertainty region using MC dropout for “F-7” signal.

Fig. 5. Uncertainty region using MC dropout for “P-1” signal.

Fig. 6. Normalized measurement scores v.s. $N_{max}$ for signal “P-1” to show the postprocessing effect.

TABLE II

MSE Comparison of Forecasting-Based Approaches

| Baseline | Bayesian LSTM | Arima [10] | LSTM [11] |
|----------|---------------|------------|------------|
| SMAP     | 0.05          | 0.61       | 0.07       |
| MSL      | 0.21          | 0.83       | 0.29       |

MSE considered in the forecasting-based approaches. Due to the fundamental difference in definitions, we have not compared the mentioned methods with our approach.

2) Recall: This is often used as a sensitivity metric. It is the proportion of relevant instances that were retrieved, i.e., $\theta = \frac{TP}{TP + FN}$, where $TP$, $FP$, $FN$, and $TN$ stand for true positive, false positive, false negative, and true negative, in the confusion matrix, respectively.

3) Precision: It is the fraction of relevant instances among the retrieved instances, i.e., $\xi = \frac{TP}{TP + FP}$.

4) Accuracy: It is one of the important classification performance measures. Based on the definition, it is the proportion of correct predictions (both true positives and negatives) among the total number of explored cases, i.e., $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$.

5) $F_1$ Score: This score is defined as the harmonic mean of precision and recall, i.e., $F_1 = \left( \frac{1}{\xi} + \frac{1}{\theta} \right)^{-1}$.

For finding the best value of $N_{max}$ in postprocessing, we perform a grid search among different values of $N_{max}$ for each signal independently and pick the optimal value in such a way that the dissimilarity between the predicted labels and the known labels is minimized. The value of $N_{max}$ depends on the maximum allowable delay, and selecting a higher value for $N_{max}$ causes a smaller value for $FP$. Alternatively, choosing a high value for $N_{max}$ (e.g., 100) leads to a high value for $FN$. Therefore, finding the optimal value for this parameter will boost the AD overall performance.

Assume a new metric $\rho_n = TP_n + TN_n - FP_n - FN_n$. We choose $N_{max}$ in such a way that $\rho_n$ gets maximized within a reasonable range of $n$. Fig. 6 shows the normalized value of different measurement criteria versus $N_{max}$ for signal “P-1.” This figure shows that after a specific amount of $N_{max}$, the value of $\rho_n$ (and other evaluation criteria) does not change. For this specific signal, the optimal value for $N_{max}$ is $N_{max}^{opt} = 8$. Also, we found that the anomalies cannot be detected using some signals (e.g., “M6” in the MSL dataset has a value of $-1$ in all training and testing phases, which makes it impossible to be used in univariate AD cases). We have excluded these types of signals (i.e., “M6, E3, A1, D1, D3, D4, G1, D5, D11, G6, R1, A6, F3, M2, P10, M3, D16, P15, P11, P14”) in our comparison results.

In Table III, we compare the $F_1$ score for the aforementioned methods for two different baselines. As the comparison results show, our proposed method outperforms the existing methods for AD. The closest methods to our
The proposed NN-based method provides a generic low-complexity, and prediction-based solution to advancing AD. Our method improves the MSE, precision, recall, accuracy, and $F_1$ score compared to existing methods. The techniques proposed for LSTM can be considered a generic time series analytics tool for autonomous FDIR tasks of satellite missions. Equally important, the proposed method addresses a critical challenge in FDIR: how to efficiently use the time series data without labels to facilitate FDIR missions. The proposed work enables efficient utilization and analysis of the time series data for satellite telemetry and other FDIR applications where on-board hardware and software AD is needed.

VI. CONCLUSION

TABLE III

| Method | Parameters | Training Time (per epoch) | Throughput |
|--------|------------|---------------------------|------------|
| Bayesian LSTM | 223902 | 1.48 s | 0.37 × 10^4 predictions/s |
| TadGan [1] | 447180 | 78 s | 0.47 × 10^4 predictions/s |
| Arima [10] | 17 | 1.1 s | 2.1 × 10^4 predictions/s |
| LSTM AE [7] | 86350 | 1.8 s | 1.16 × 10^4 predictions/s |
| LSTM [11] | 228009 | 1.4 s | 2.11 × 10^4 predictions/s |
| MCD-BiLSTM-VAE [5] | 550993 | 6.59 s | 0.012 × 10^4 predictions/s |

TABLE IV

| Methods | Parameters | Training Time (per epoch) | Throughput |
|--------|------------|---------------------------|------------|
| Bayesian LSTM | 0.79 | 0.82 | 0.92 |
| TadGan [1] | 0.76 | 0.76 | 0.87 |
| Arima [10] | 0.76 | 0.81 | 0.97 |
| LSTM AE [7] | 0.76 | 0.76 | 0.87 |
| LSTM [11] | 0.63 | 0.56 | 0.75 |
| MCD-BiLSTM-VAE [5] | 0.65 | 0.58 | 0.76 |

TABLE V

| Methods | SSMAF | MSL |
|--------|-------|-----|
| Training | Recall | Precision | Accuracy | Recall | Precision | Accuracy |
| Bayesian LSTM | 0.79 | 0.82 | 0.92 | 0.97 | 0.94 | 0.99 |
| TadGan [1] | 0.76 | 0.76 | 0.87 | 0.78 | 0.81 | 0.85 |
| Arima [10] | 0.76 | 0.81 | 0.97 | 0.80 | 0.83 | 0.86 |
| LSTM AE [7] | 0.76 | 0.76 | 0.87 | 0.63 | 0.56 | 0.75 |
| LSTM [11] | 0.63 | 0.58 | 0.76 | 0.60 | 0.58 | 0.74 |

REFERENCES

[1] J. Chen et al., “Imbalanced satellite telemetry data anomaly detection model based on Bayesian LSTM,” Acta Astronautica, vol. 180, pp. 232–242, 2021.
[2] E.-H. Li et al., “Intelligent analysis algorithm for satellite health under time-varying and extremely high thermal loads,” Entropy, vol. 21, no. 10, 2019, Art. no. 983.
[3] H. Safaeipour, M. Forouzanfar, and A. Casavola, “A survey and classification of incipient fault diagnosis approaches,” J. Process Control, vol. 97, pp. 1–16, 2021.
[4] D. Cayrac, D. Dubois, and H. Prade, “Handling uncertainty with possibility theory and fuzzy sets in a satellite fault diagnosis application,” IEEE Trans. Fuzzy Syst., vol. 4, no. 3, pp. 251–269, Aug. 1996.
[5] M. Schwabacher, N. Oza, and B. Matthews, “Unsupervised anomaly detection for liquid-fueled rocket propulsion health monitoring,” J. Aerosp. Comput. Inf. Commun., vol. 6, no. 7, pp. 464–482, 2009.
[6] S. K. Ibrahim et al., “Machine learning techniques for satellite fault diagnosis,” Ain Shams Eng. J., vol. 11, no. 1, pp. 45–56, 2020.
[7] K. Hundman et al., “Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 387–395.
[8] N. Ding et al., “Multivariate-time-series-driven real-time anomaly detection based on Bayesian network,” Sensors, vol. 18, no. 10, 2018, Art. no. 3367.
[9] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, 2009.
[10] E. H. M. Pena, M. V. O. de Assis, and M. L. Procenca, “Anomaly detection using forecasting methods ARIMA and HWDS,” in Proc. 32nd Int. Conf. Chilean Comput. Sci. Soc., 2013, pp. 63–66.
[11] I. Tinawi, “Machine learning for time series anomaly detection,” Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2019.
[12] P. Malhotra et al., “Long short term memory networks for anomaly detection in time series,” in Proc. 23rd Eur. Symp. Artif. Neural Netw., 2015. [Online]. Available: https://www.esann.org/sites/default/files/proceedings/legacy/es2015-56.pdf.
[13] S. Ahmad et al., “Unsupervised real-time anomaly detection for streaming data,” Neurocomputing, vol. 262, pp. 134–147, 2017.

CORRESPONDENCE
D. Salinas et al., “DeepAR: Probabilistic forecasting with autoregressive recurrent networks,” *Int. J. Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

Y. Su et al., “Robust anomaly detection for multivariate time series through stochastic recurrent neural network,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2828–2837.

A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, “TadGAN: Time series anomaly detection using generative adversarial networks,” in *Proc. IEEE Int. Conf. Big Data*, 2020, pp. 33–43.

D. Li et al., “MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks,” in *Proc. Int. Conf. Artif. Neural Netw.*, 2019, pp. 703–716.

S. Liu et al., “Time series anomaly detection with adversarial reconstruction networks,” *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2021.3140058.

T. Choi, D. Lee, Y. Jung, and H.-J. Choi, “Multivariate time-series anomaly detection using SeqVAE-CNN hybrid model,” in *Proc. Int. Conf. Inf. Netw.*, 2022, pp. 250–253.

K.-J. Jeong and Y.-M. Shin, “Time-series anomaly detection with implicit neural representation,” 2022, arXiv:2201.11950.

S. Tuli, G. Casale, and N. R. Jennings, “TranAD: Deep transformer networks for anomaly detection in multivariate time series data,” 2022, arXiv:2201.07284.

L. Yang et al., “Improved deep learning based telemetry data anomaly detection to enhance spacecraft operation reliability,” *Microelectronics Rel.*, vol. 126, 2021, Art. no. 114311.

A. Loquercio, M. Segu, and D. Scaramuzza, “A general framework for uncertainty estimation in deep learning,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3153–3160, Apr. 2020.

Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

A. Graves, “Practical variational inference for neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 2348–2356.

T. Bui et al., “Deep Gaussian processes for regression using approximate expectation propagation,” in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1472–1481.

D. T. Mirikitani and N. Nikolaev, “Dynamic modeling with ensemble Kalman filter trained recurrent neural networks,” in *Proc. 7th Int. Conf. Mach. Learn. Appl.*, 2008, pp. 843–848.

M. A. M. Sadr, J. Gante, B. Champagne, G. Falcao, and L. Sousa, “Uncertainty estimation via Monte Carlo dropout in CNN-based mmWave MIMO localization,” *IEEE Signal Process. Lett.*, vol. 29, pp. 269–273, 2022, doi: 10.1109/LSP.2021.3130504.

Y. Gal and Z. Ghahramani, “A theoretically grounded application of dropout in recurrent neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1019–1027.

L. Mou et al., “A multi-scale anomaly detection framework for retinal OCT images based on the Bayesian neural network,” *Biomed. Signal Process. Control*, vol. 75, 2022, Art. no. 103619. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1746809422001410

J. Zgraggen, G. Pizza, and L. G. Huber, “Uncertainty informed anomaly scores with deep learning: Robust fault detection with limited data,” in *Proc. PHM Soc. Eur. Conf.*, 2022, pp. 530–540.

D. Codetta-Raiteri and L. Portinale, “Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 1, pp. 13–24, Jan. 2015.

W. Gao and Z.-H. Zhou, “Dropout Rademacher complexity of deep neural networks,” *Sci. China Inf. Sci.*, vol. 59, no. 7, pp. 1–12, 2016.

A. Bifet and R. Gavalda, “Learning from time-changing data with adaptive windowing,” in *Proc. SIAM Int. Conf. Data Mining*, 2007, pp. 443–448.

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

M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.