An Anomaly Detection Method for Satellites Using Monte Carlo Dropout

Mohammad Amin Maleki Sadr, Yeying Zhu, *Member, IEEE,* Peng Hu, *Senior Member, IEEE*

***Abstract*—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 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 paper, we present a tractable approximation for BNN based on the Monte Carlo (MC) dropout method for capturing the uncertainty in a 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 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 post-processing 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.**

***Index Terms*—Anomaly detection, telemetry time series data, satellite communications, uncertainty estimation.**

I. Introduction

Satellites are complex systems composed of many parts that are interrelated and mutually restricted. They are a fusion of multi-disciplinary technologies such as telemetry sensing, mobile communications, and navigation systems. Proactive diagnosis of failures, anomaly detection (AD), and response to potential hazards are required to guarantee the availability and continuity of satellite services [1]. Considering the complex design structure and harsh environment of space, AD on satellites cannot be performed directly in outer space; instead, the telemetry data is employed to provide healthy key information that can be used to recover the satellite from possible problems [2]. 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 intelligent AD methods, which are data-based, are recommended [3].

Many studies focused on deep neural networks (DNN) for data-based AD for spacecraft applications [4], [5]. AD in this context has been studied from two different standpoints, which are:

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 deviations from the expected response can be flagged as a tentative anomaly [6]–[10].

2) Reconstruction-based method: In this AD method, time series data is reconstructed, and then by comparing the real (true) and reconstructed values, anomalies are detected [11]–[14]. As this approach must reconstruct the time series to detect anomalies, it cannot be performed in real-time. The OmniAnomaly method in [12] 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, generative adversarial networks (GANs) for time series AD are proposed in [13]. In [11], the performance of Long Short-Term Memory (LSTM) for AD for the space shuttle dataset is discussed. In [6], the LSTM Auto-Encoder (AE) efficiency for detecting satellite anomalies is demonstrated.

Although both of these methods can capture the anomaly points, knowing the confidence of an NN model could be critical. More specifically, model uncertainty is essential for assessing how much to rely on the forecast produced by the model and it plays an important role in applications like AD. The uncertainty measure can be the variance or confidence intervals around the prediction made by the 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) [15]. As BNN does not suffer from over-fitting and always offers uncertainty estimation, it grants a significant amount of interest in many applications [16]. Finding the posterior distribution of BNN is challenging and computationally intractable. Therefore, the posterior distribution needs to be approximated with different techniques [16]–[20]. MC dropout has demonstrated important benefits, such as lower computational cost and higher precision, over other methods [16]. The usage of MC dropout for RNN (i.e., LSTM) and the mathematical formulation can be found in [21]. In [22], the authors have proposed an AD technique based on an approximation of BNN.

In this paper, we use MC dropout to capture the con-

The work was supported by the National Research Council Canada under Grant No. CH-HTSN-418. Mohammad Amin Maleki Sadr and Yeying Zhu are with the Dept. of Statistics and Actuarial Science, University of Waterloo, ON N2L 3G1, Canada. Peng Hu is with Digital Technologies Research Centre, National Research Council Canada, Waterloo, ON N2L 3G1, Canada, and with Dept. of Statistics and Actuarial Science, University of Waterloo, ON N2L 3G1, Canada. (Peng.Hu@nrc-cnrc.gc.ca)

fidence interval for the NASA satellite telemetry dataset. Specifically, by using MC dropout, we derive the first- and second-order statistics of an NN consisting of the LSTM layers 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. To be noted, our AD method is a forecasting-based method. Furthermore, the contributions of this paper are summarized as follows:

- In our pre-processing method, unlike the standard pre-processing steps such as normalization and data cleaning, which are implemented in [6], [10], [13], [14], [22], we first clean, normalize and scale the dataset, then use an innovative algorithm, which performs an adaptive weighted averaging inside a variable length window to smooth the time series. To the best of our knowledge, no prior work can be found that uses this method.
- In contrast to the reconstruction-based methods in some studies such as [6], [13], [22], which must utilize all the samples of the time series for AD (and cannot be applied in real-time), our AD approach only uses a small portion of the time series to detect an anomaly.
- Our approximation of BNN is more precise than the method in [22]. The MCD-BiLSTM-VAE method in [22], uses dropout only before the dense layers rather than the entire network composed of both the LSTM cells and the dense layers. It only captures the uncertainty of the dense layers. In contrast, we applied MC dropout at both 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 [16], [21].
- Different from the methods in [6], [10], [13], [14], which might suffer from over-fitting, our method uses the dropout at each layer separately. By doing this, not only reduces the complexity but also by adding a regularization term, prevents the model from being over-fitted [23].
- Different from the approaches in [6], [13], [22], which suffer from a high number of false positives, we propose a post-processing algorithm which is based on the divergence of a succession of predicted data points from the corresponding confidence intervals. This method lessen the number of false positives.
- Unlike the thresholding approach in [22], which is static, we propose a dynamic thresholding method. For the dataset under consideration, the dynamic thresholding method surpasses the approach of [22] in terms of more accurate prediction and better classification results.
- We have compared the performance of all existing methods with our proposed Bayesian approach. As the results of the comparison show[1], our method outperforms the various competitors (considering the trade-off between complexity and performance measures).

[1]Simulation codes and results of our study can be found at https://github.com/DreamweaverU/SatAnomalyDetection

The rest of this paper is organized as follows: in Section II, we discuss the proposed pre-processing method. Bayesian LSTM methods for training and post-processing AD filters are discussed in Section III. Performance evaluation of the proposed AD method is presented in Section IV followed by concluding remarks in Section V.

II. Pre-processing for anomaly detection

The raw telemetry data could not be directly ingested into our AD algorithm, and it should first go through a pre-processing stage [6]. The weighted moving average (WMA) approach is adopted to improve the prediction accuracy[2] by making the time series smoother [24]. However, when we have anomaly points in the time series, the standard WMA may not be applicable. In fact, due to the averaging operation between the current point and the adjacent points, there exists a high chance of missing an anomaly point. Our motivation here is to devise a strategy to take advantage of the benefits of WMA for improving the prediction accuracy without missing any anomaly point. We propose an averaging from 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, so a narrower window length should be used, as we do not want the anomaly point to be missed before applying the anomaly detection approach. Alternatively, when the difference between two consecutive points is small, we choose a wider window to make the time series smoother. Assume that $\mathbf{u} \in \mathbb{R}^{n\times 1}$ and $\mathbf{x} \in \mathbb{R}^{n\times 1}$ are the raw data and the pre-processed data, respectively. Let $l_w(k)$ be the length of kth window of our filter. Assume that the processed data in the kth window is $\mathbf{x}(k)$, where $k \in \mathcal{J}, \mathcal{J} = \{1, ..., n\}$. The smoothed data is given by:

$$\mathbf{x}(k) = \frac{1}{\|\kappa\|} \sum_{i=-\frac{l_w(k)}{2}}^{\frac{l_w(k)}{2}} \alpha_i \mathbf{u}(k-i), \tag{1}$$

where $\kappa = [\ \alpha_{\frac{-l_w(k)}{2}} \ \cdots \ \alpha_{\frac{l_w(k)}{2}}\ ]^T$, $\|.\|$ is the vector norm function, and $\alpha_i, i \in \mathcal{I} = \left\{\frac{-l_w(k)}{2}, ..., \frac{l_w(k)}{2}\right\}$ is the smoothed filter coefficients. In our approach, the smoothing coefficients $0 \leq \alpha_i \leq 1$, depend on a distance function $\mathbf{d}$, where $\mathbf{d}(k-i) = |\mathbf{u}(k-i-1) - \mathbf{u}(k-i)|$. The distance function is the absolute difference between the two successive data points inside the kth window. In this pre-processing algorithm, we start with $l_w(0) = 2$ and $\alpha_0 = 1$. Let the distance threshold[3] 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. For the ith data

[2]In our terminology, the high prediction accuracy is equivalent to low mean square error (MSE) between the real values and the predicted values.

[3] Note that we choose the value of the threshold $d_{th}^{(k)} = m^{(k)} + 2\sigma^{(k)}$ 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.

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

$$\alpha_i = \begin{cases} 0 & i \in \mathcal{I} - \{0\} \\ 1 & i = 0, \end{cases} \tag{2}$$

then we increment k, and proceed to another window i.e., $k = k+1$. The pre-processing procedure is shown in Algorithm 1.

Algorithm 1 Pre-processing algorithm

```
     Input: $\mathbf{u}(k)$,
     Output: $\mathbf{x}(k)$
 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 $k$th windows with length $l_w(k)$
 4:     Select the $k$th threshold as $d_{th}^{(k)} = m^{(k)} + 2\sigma^{(k)}$
 5:     for $i \in \mathcal{I}$ do
 6:         if  $\mathbf{d}(k-i) < d_{th}^{(k)}$  then
 7:             $\alpha_i = 1$
 8:             $l_w(k) = l_w(k) + 1$
 9:         else if  $\mathbf{d}(k-i) > d_{th}^{(k)}$  then
10:             Break;
11:         end if
12:     end for
13:     Calculate $\mathbf{x}(k) = \frac{1}{\|\kappa\|} \sum_{i=-\frac{l_w(k)}{2}}^{\frac{l_w(k)}{2}} \alpha_i \mathbf{u}(k-i)$
14:     Go to the next window, i.e., $k = k+1$
15: end while
```

III. 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 layers followed by dense layers. In this section, we adopt MC dropout for approximation of the BNN model in both 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 $\mathbf{x} \in \mathbb{R}^T$ and $\mathbf{y} \in \mathbb{R}^{T_D}$ as the input and output of NN, respectively. Assume that $\boldsymbol{\omega}$ with prior distribution $p(\boldsymbol{\omega})$ represents a collection of the NN parameters. The prediction of the output vector for an associated input vector $\mathbf{x}$ is:

$$p(\mathbf{y}\,|\mathbf{x}, \mathbf{D}) = \mathrm{E}_{p(\boldsymbol{\omega}|\mathbf{D})}\left(p(\mathbf{y}\,|\mathbf{x}, \boldsymbol{\omega})\right), \tag{3}$$

where $\mathbf{D} = \{(x_t, y_t)\ \forall t \in \{1, ..., T\}\}$ and $p(\boldsymbol{\omega}|\mathbf{D})$ are a batch of data, and the posterior distribution on weights,

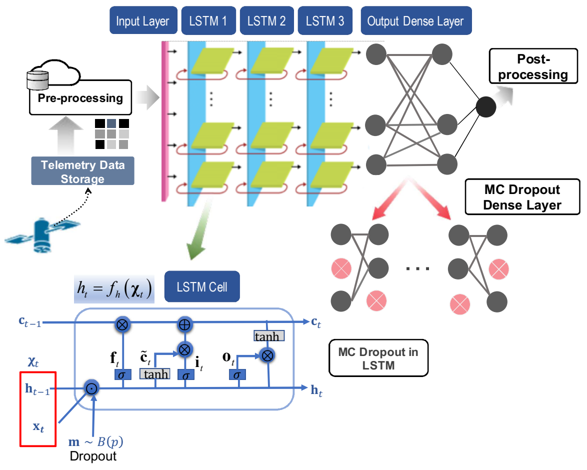

Figure 1: An illustration of the proposed system model using Bayesian LSTM

respectively. The expectation under $p(\boldsymbol{\omega}|\mathbf{D})$ 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 $p(\boldsymbol{\omega}\,|\mathbf{D})$ with a simpler distribution $q(\boldsymbol{\omega})$. A metric of similarity (or distance) between two probability distribution functions (PDF) is Kullback-Leibler (KL) divergence. Here, we minimize the $\mathrm{KL}(q(\boldsymbol{\omega})\,\|p(\boldsymbol{\omega}\,|\mathbf{D}))$ which is the KL distance between $p(\boldsymbol{\omega}\,|\mathbf{D})$ and $q(\boldsymbol{\omega})$ [16]. More specifically, we solve the following problem:

$$\min_{q(\omega)} \quad \mathrm{KL}(q(\omega)\,\|p(\omega\,|\mathbf{D})), \tag{4}$$

which is equal to minimizing the following:

$$\min_{q(\omega)} -\int q(\omega) \log p(\mathbf{y}\,|\mathbf{x}\,, \omega) d\omega + \mathrm{KL}(q(\omega)\|p(\omega)), \tag{5}$$

where $p(\boldsymbol{\omega})$ 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 over-fitted. Note that minimizing the second component is equivalent to finding a $q(\omega)$ close to the prior, which essentially avoids over-fitting. The first component, on the other hand, can be rewritten with the MC sampling over $\boldsymbol{\omega}$ with a single sample as the following:

$$-\frac{1}{T} \sum_{n=1}^{T} \int \log p(\mathbf{y}_n | f^{\boldsymbol{\omega}}(\mathbf{x}_n)) d\boldsymbol{\omega}. \tag{6}$$

In the next subsection, we will discuss the detailed structure of the function $f^{\boldsymbol{\omega}}(\mathbf{x}_n)$ 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 various other control gates. The following formulation briefly shows the principle of an LSTM [25]:

$$\begin{aligned}&\mathbf{i}_t=\sigma\left(\mathbf{W}_i\chi_t+\mathbf{b}_i\right), \quad \mathbf{f}_t=\sigma\left(\mathbf{W}_f\chi_t+\mathbf{b}_f\right),\\ &\mathbf{o}_t=\sigma\left(\mathbf{W}_o\chi_t+\mathbf{b}_o\right), \quad \tilde{\mathbf{c}}_t=\tanh\left(\mathbf{W}_{\tilde{c}}\boldsymbol{\chi}_t+\mathbf{b}_c\right),\\ &\mathbf{c}_t=\mathbf{f}_t\odot\mathbf{c}_{t-1}+\mathbf{i}_t\odot\tilde{\mathbf{c}}_t, \quad \mathbf{h}_t=\mathbf{o}_t\odot\tanh(\mathbf{c}_t),\end{aligned} \tag{7}$$

where $\boldsymbol{\chi}_t = \begin{bmatrix} \mathbf{x}_t & \mathbf{h}_{t-1} \end{bmatrix}^T$ and $\mathbf{x}_t$ represents the input data, σ is the logistic sigmoid function and $\odot$ is the element-wise product. Also, $\mathbf{c}_t$ represents the state value of an LSTM unit (while $\tilde{\mathbf{c}}_t$ represents the candidate state value), $\mathbf{i}_t$ represents the state value of the input gate, $\mathbf{f}_t$ represents the state value of the forget gate, $\mathbf{o}_t$ represents the output state value of the output gate, $\mathbf{h}_t$ represents the output of the LSTM unit, and t represents the current time step. $\mathbf{W}$ and $\mathbf{b}$ 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):

$$\mathbf{h}_t = f_h(\boldsymbol{\chi_t}). \tag{8}$$

By applying a dropout at the input and hidden layers, we have [21]:

$$\mathbf{i}_t=\sigma\left(\mathbf{W}_i(\boldsymbol{\chi}_t\odot\mathbf{m})+\mathbf{b}_i\right), \tag{9}$$
$$\mathbf{f}_t=\sigma\left(\mathbf{W}_f(\boldsymbol{\chi}_t\odot\mathbf{m})+\mathbf{b}_f\right), \tag{10}$$
$$\mathbf{o}_t=\sigma\left(\mathbf{W}_o(\boldsymbol{\chi}_t\odot\mathbf{m})+\mathbf{b}_o\right), \tag{11}$$
$$\tilde{\mathbf{c}}_t=\tanh\left(\mathbf{W}_{\tilde{c}}(\boldsymbol{\chi}_t\odot\mathbf{m})+\mathbf{b}_c\right), \tag{12}$$
$$\mathbf{c}_t=\mathbf{f}_t\odot\mathbf{c}_{t-1}+\mathbf{i}_t\odot\tilde{\mathbf{c}}_t, \tag{13}$$

where $\mathbf{m}\sim\mathcal{B}(p)$ is a random mask vector where each row has a Bernoulli distribution with the parameter of p and it is repeated at all time steps. We use the MC dropout in the dense layers by dropping the neurons randomly using a Bernoulli distribution at the testing phase. For the kth dense layer, we have [16]:

$$\begin{aligned}&\mathbf{W}_k=\mathrm{diag}([z_{k,j}]_{j=1}^{K_i})\mathbf{M}_k,\\ &z_{k,j}\sim\mathcal{B}\left(p_k\right) \quad \forall k\in\{1,...,D\}\,,\,j\in\{1,...,K_{k-1}\},\end{aligned} \tag{14}$$

where p_k and matrix $\mathbf{M}_k$ 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 layer k. By assuming $\mathbf{b}_k$ as a bias at the kth dense layer, the weight matrix (NN parameter) can be considered as $\boldsymbol{\omega}=[\mathbf{W}_k,\mathbf{W}_i,\mathbf{W}_f,\mathbf{W}_o,\mathbf{W}_{\tilde{c}},\mathbf{b}_k,\mathbf{b}_i,\mathbf{b}_f,\mathbf{b}_o,\mathbf{b}_c]$. Therefore, (6) is rewritten as

$$\sum_{i=1}^{N}\log(p(y_i|\sqrt{\tfrac{1}{K_D}}\hat{\mathbf{W}}_D\sigma(...\sqrt{\tfrac{1}{K_2}}\hat{\mathbf{W}}_2\sigma(\hat{\mathbf{W}}_1(\boldsymbol{\psi_T})+\mathbf{b}_1)...))), \tag{15}$$

where $\boldsymbol{\psi}_T\triangleq[\mathbf{x}_{T,i},f_h^{\hat{\boldsymbol{\omega}}}(...f_h^{\hat{\boldsymbol{\omega}}}([\mathbf{x}_{0,i},\mathbf{h}_0])]$, and $\hat{\boldsymbol{\omega}}\sim q\left(\boldsymbol{\omega}\right)$. Note that for each sequence $\mathbf{x}_i$, we sample a new realization $\hat{\boldsymbol{\omega}}=[\hat{\mathbf{W}}_k,\hat{\mathbf{W}}_i,\hat{\mathbf{W}}_f,\hat{\mathbf{W}}_o,\hat{\mathbf{W}}_{\tilde{c}},\hat{\mathbf{b}}_k,\hat{\mathbf{b}}_i,\hat{\mathbf{b}}_f,\hat{\mathbf{b}}_o,\hat{\mathbf{b}}_c]$. Each symbol in the sequence $\mathbf{x}_i=[\mathbf{x}_{i,1},...,\mathbf{x}_{i,T}]$ is passed through the function $f_h^{\hat{\boldsymbol{\omega}}}$ with the same weight realizations $\hat{\boldsymbol{\omega}}$ used at every time step $t\leq T$. In the dense layers, and LSTM layers, the MC dropout is performed during both 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.

Algorithm 2 MC dropout algorithm

```
Repeat:
Sample $z_{i,j}\sim\mathcal{B}(p_i)$, $i\in\{1,...,D\}$, $j\in\{1,...,K_{i-1}\}$, and $\mathbf{m}\sim\mathcal{B}(p_k)$ $\forall k\in\{1,...,L\}$ and set (12), (13), (14), (15), (16), (17), respectively, and find $\hat{\boldsymbol{\omega}}=[\hat{\mathbf{W}}_k,\hat{\mathbf{W}}_i,\hat{\mathbf{W}}_f,\hat{\mathbf{W}}_o,\hat{\mathbf{W}}_{\tilde{c}},\hat{\mathbf{b}}_k,\hat{\mathbf{b}}_i,\hat{\mathbf{b}}_f,\hat{\mathbf{b}}_o,\hat{\mathbf{b}}_c]$ where $\hat{\boldsymbol{\omega}}\sim q(\boldsymbol{\omega})$
Minimize (one step):
    $-\int q(\boldsymbol{\omega})\log\left(p\left(\mathbf{y}\left|\mathbf{x},\,\boldsymbol{\omega}\right.\right)\right)+\mathrm{KL}\left(q\left(\boldsymbol{\omega}\right),p\left(\boldsymbol{\omega}\right)\right)$
```

Assume that we have l sets of realization of NN after applying the MC dropout as shown in Fig. 1. As derived in [16], the approximated predictive distribution is given by the following:

$$q(\mathbf{y}\left|\mathbf{x}\right.)=\int p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega})q(\boldsymbol{\omega})d\boldsymbol{\omega} \tag{16}$$

The first two moments are derived as [16]:

$$\mathrm{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})\approx\frac{1}{l}\sum_{t=1}^{l}\mathrm{E}(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}^t) \tag{17}$$

and

$$\mathrm{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y}^T\mathbf{y})\approx\frac{1}{l}\sum_{t=1}^{l}\mathrm{E}(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}^t)^T\mathrm{E}(\mathbf{y}|\mathbf{x},\boldsymbol{\omega}^t) \tag{18}$$

The model's predictive variance is

$$\mathrm{Var}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})=\mathrm{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y}^T\mathbf{y})-\mathrm{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})^T\mathrm{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y}) \tag{19}$$

C. Post-processing of AD

To reduce the number of false positives, we propose a post-processing 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 a tentative anomaly point. In our specific dataset, anomaly points do not occur as a single point, but they emerge as a sequence (burst) of dependent points in a particular part of the time series.

It is important to determine the time when the anomaly starts. Let N_{max} be a range of consecutive data points in time series. In our approach, if $0.8N_{max}$ 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 post-processing approach can significantly reduce the number of false positives. Note that choosing a suitable value for N_{max} depends on the maximum allowed delay. We can improve the accuracy if we wait for more sample points to announce the anomaly; however, no successful alert with a huge delay is useful. Therefore, if an AD algorithm raises an alert fast enough (i.e., before a maximum allowed delay) the whole anomaly fragment will be considered to be detected successfully. We will further illustrate this method in numerical studies.

IV. Numerical Results

First, we describe the methodology for evaluating AD, then summarize the specific dataset under consideration, followed by the performance evaluation.

A. Methodology

We assume an NN with 3 LSTM layers 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 [6] to compare and evaluate our method with the following approaches: i) dynamic thresholding approach with LSTM Auto-Encoder (AE) [6]; ii) TadGan approach [13]; iii) MadGan method [14]; iv) Arima method [9]; v) LSTM method [10]; and vi) MCD-BiLSTM-VAE method in [22]. We will use the following metrics to evaluate the performance of our model: mean squared error (MSE), F_1 score, accuracy, recall, and precision. Moreover, we will compare the computational complexity of different methods. These metrics have been widely used in the literature and can be found in details in [26].

B. NASA dataset

We use the satellite telemetry dataset, which was originally collected by NASA in [6]. The dataset comes from two spacecrafts: the Soil Moisture Active Passive (SMAP) satellite and the Curiosity Rover on Mars (MSL). There are 82 signals available in the NASA dataset, which are detailed in Table I. We found 54 of the 82 signals (multivariate time series) are continuous, and the remaining signals 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 row of the timestamped 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.

Table I: NASA telemetry dataset summary

Dataset	Signals	Anomaly Sequences	Data Points	Anomaly Points
SMAP	55	67	562800	54696
MSL	27	36	132046	7766

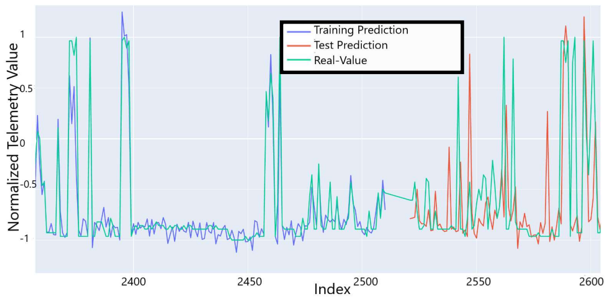

Figure 2: Examples of real values and predictions in training and testing modes for 'F-7' signal

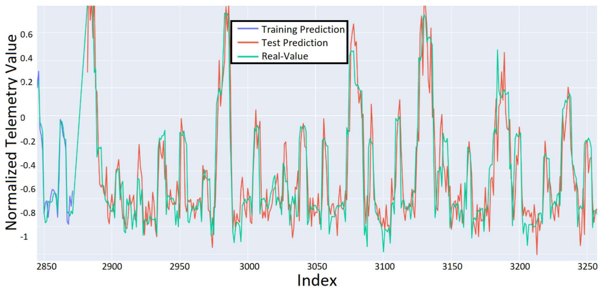

Figure 3: Examples of real values and predictions in training and testing modes for 'P-1' signal

C. Evaluation

In this subsection, we evaluate and compare the proposed model with those of other studies. Let us first show 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'. These results are shown in Fig. 2 and Fig. 3, respectively. By comparing these two figures, we notice that the SMAP data performs better in the prediction phase. To show the uncertainty region by our Bayesian approach, Fig. 4 and Fig. 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% of the predicted values are inside the uncertainty bounds. Furthermore, we use the following evaluation measures to assess our method:

1) Mean Squared Error: MSE, which is used as a metric for comparing forecasting-based methods, assesses the average squared difference between the real and predicted values. It is defined as $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2$, where $\hat{\mathbf{y}}_i$ is the predicted value at the ith timestamp, i.e., $\mathrm{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})$. The numerical results for the MSE of two datasets for our approach, the Arima method [9], and the LSTM method [10] can be found in Table II. Note that the MSE can be used just for forecasting-based methods, and for the reconstruction-based methods such as TadGan and MadGan, LSTM AE, it cannot be calculated [4].

Table II: MSE comparison of forecasting-based approaches

Baseline	Bayesian LSTM	Arima [9]	LSTM [10]
SMAP	0.06	0.68	0.09
MSL	0.17	0.83	0.23

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., $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$

5) F_1 score: This score is defined as the harmonic mean of precision and recall, i.e., $F_1 = \left(\frac{\xi^{-1}+\theta^{-1}}{2}\right)^{-1}$.

Before discussing the numerical comparison of different approaches, let us discover how to find the best value for N_{max} in post-processing. 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. Note that, the value of N_{max} depends on the maximum-allowed 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. Let us define a new metric $\rho_n = TP_n + TN_n - FP_n - FN_n$. We choose $N_{\max}$ in such a way that ρ_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 ρ_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 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 uni-variate 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.

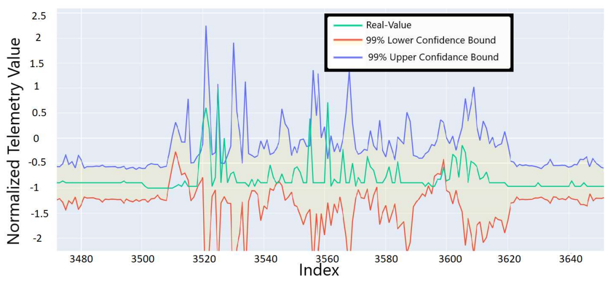

Figure 4: Uncertainty region using MC dropout for 'F-7' signal

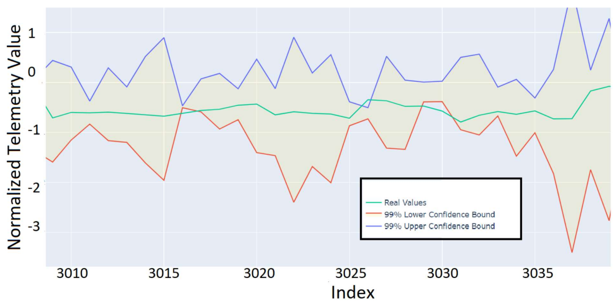

Figure 5: Uncertainty region using MC dropout for 'P-1' signal

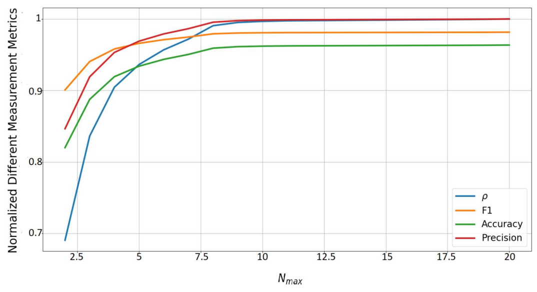

Figure 6: The normalized measurement scores v.s. N_{max} for signal 'P-1' to show the post-processing effect.

Table III: F1 score comparison

Baseline	Bayesian LSTM	TadGan [13]	Arima [9]	LSTM AE [6]	MadGan [14]	LSTM [10]	MCD-BiLSTM-VAE [22]
SMAP	0.84	0.70	0.42	0.69	0.12	0.62	0.67
MSL	0.69	0.62	0.49	0.55	0.11	0.48	0.56

Table IV: Comparison of learnable parameters, training time, and inference throughput

Methods	Parameters	Training Time (per epoch)	Throughput
Bayesian LSTM	229409	4.83 sec	0.57×10^3 predictions/sec
TadGan [13]	447189	32 sec	2.51×10^2 predictions/sec
Arima [9]	48209	1.1 sec	1.28×10^3 predictions/sec
LSTM AE [6]	86250	1.84 sec	1.16×10^3 predictions/sec
LSTM [10]	229409	4.83 sec	1.36×10^3 predictions/sec
MCD-BiLSTM-VAE [22]	580993	6.96 sec	0.41×10^3 predictions/sec

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 method in the F_1 score are TadGan, LSTM AE and MCD-BiLSTM-

[4]Authors in [6], [13], [14], [22], used the MSE terminology to refer to 'reconstruction-error'. Indeed, what they calculated is the quantity of error for reconstructing the time series and it is different from the concept of MSE considered in this paper. Due to the fundamental difference in definitions, we do not compare the mentioned methods with our approach.

Table V: Comparison of accuracy, precision, and recall

Methods	**SMAP**			**MSL**		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Bayesian LSTM	**0.79**	**0.84**	**0.82**	**0.7**	**0.74**	**0.84**
TadGan [13]	0.76	0.76	0.64	0.58	0.58	0.56
Arima [9]	0.52	0.61	0.57	0.49	0.53	0.49
LSTM AE [6]	0.73	0.76	0.79	0.55	0.65	0.66
LSTM [10]	0.67	0.56	0.73	0.53	0.48	0.62
MCD-BiLSTM-VAE [22]	0.71	0.77	0.78	0.44	0.51	0.64

VAE. The difference between our method and MadGan approach is significant; for this reason, we have excluded the MadGan approach from the rest of the comparisons. In addition to F_1 score improvement, our method still has other advantages over TadGan, LSTM AE, LSTM, and MCD-BiLSTM-VAE approaches, which have been discussed in Section I. One of the distinct advantages is the complexity reduction of our Bayesian approach compared to some of methods (e.g., TadGan). Let us examine the complexity cost of different methods in details.

Table IV shows the computational cost of our proposed approach compared with the other competing methods. The reported results are performed using the NVIDIA GeForce GTX 1650 GPU, 16 GB DDR4-3200 MHz RAM, and PyTorch framework. As the comparison results show, the Arima method has the best performance from the computational complexity viewpoint. While our method significantly outperforms the TadGan approach from the computational perspective, the LSTM AE method exceeds our method in terms of complexity. The training time of our method is identical to the LSTM approach. The reason behind this is that the MC dropout does not affect the NN during the training phase, instead during the testing phase, applying the dropout, enables us to approximate the original BNN. The complexity cost of MCD-BiLSTM-VA method is slightly higher than our approach. Furthermore, in Table V, we compare the precision, recall, and accuracy of our method with other approaches. As the comparison results show, our method outperforms all other methods in all three performance metrics.

V. Conclusions

In this paper, we proposed the MC dropout method for capturing the anomalies in satellite telemetry channels. The MC dropout method provides a predicted value and the uncertainty region around it, based on which we detect the anomaly points. Our proposed NN consists of several LSTM layers followed by the dense layers. We applied the MC dropout at both the LSTM and dense layers separately. We also proposed innovative pre-processing and post-processing procedures for AD. The numerical results showed that our proposed time series AD method outperforms the existing methods from different evaluation metric perspectives.

References

[1] E.-H. Li, Y.-Z. Li, T.-T. Li *et al.*, "Intelligent analysis algorithm for satellite health under time-varying and extremely high thermal loads," *Entropy*, vol. 21, no. 10, p. 983, 2019.

[2] 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.

[3] 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, 1996.

[4] M. Schwabacher, N. Oza, and B. Matthews, "Unsupervised anomaly detection for liquid-fueled rocket propulsion health monitoring," *J. Aeros. Comp. Inf. Com.*, vol. 6, no. 7, pp. 464–482, 2009.

[5] S. K. Ibrahim, A. Ahmed, M. A. E. Zeidan *et al.*, "Machine learning techniques for satellite fault diagnosis," *Ain Shams Engineering Journal*, vol. 11, no. 1, pp. 45–56, 2020.

[6] K. Hundman, V. Constantinou, C. Laporte *et al.*, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *24th Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2018, pp. 387–395.

[7] N. Ding, H. Gao, H. Bu *et al.*, "Multivariate-time-series-driven real-time anomaly detection based on bayesian network," *Sensors*, vol. 18, no. 10, p. 3367, 2018.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[9] E. H. Pena, M. V. de Assis, and M. L. Proença, "Anomaly detection using forecasting methods arima and hwds," in *32nd Int. Conf. Chil. Comput. Sci. Soc. SCCC (SCCC)*. IEEE, 2013, pp. 63–66.

[10] I. Tinawi, "Machine learning for time series anomaly detection," Ph.D. dissertation, Massachusetts Institute of Technology, 2019.

[11] P. Malhotra, L. Vig, G. Shroff *et al.*, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89, 2015, pp. 89–94.

[12] Y. Su, Y. Zhao, C. Niu *et al.*, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2019, pp. 2828–2837.

[13] A. Geiger, D. Liu, S. Alnegheimish *et al.*, "Tadgan: Time series anomaly detection using generative adversarial networks," in *IEEE Int. Conf. Big Data*. IEEE, 2020, pp. 33–43.

[14] D. Li, D. Chen, B. Jin *et al.*, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *Int. Conf. Artificial Neural Netw.*, 2019, pp. 703–716.

[15] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3153–3160, 2020.

[16] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, vol. 48, 2016, pp. 1050–1059.

[17] A. Graves, "Practical variational inference for neural networks," in *Adv Neural Inform Process Syst*, vol. 24, 2011, pp. 2348–2356.

[18] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato *et al.*, "Deep Gaussian processes for regression using approximate expectation propagation," in *ICML*, 2016, pp. 1472–1481.

[19] D. T. Mirikitani and N. Nikolaev, "Dynamic modeling with ensemble Kalman filter trained recurrent neural networks," in *7th ICMLA*. IEEE, 2008, pp. 843–848.

[20] M. A. Maleki Sadr, J. Gante, B. Champagne *et al.*, "Uncertainty Estimation via Monte Carlo Dropout in CNN-based mmWave MIMO Localization," *IEEE Signal Process. Lett.*, pp. 1–1, 2021.

[21] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 29, pp. 1019–1027, 2016.

[22] J. Chen, D. Pi, Z. Wu *et al.*, "Imbalanced satellite telemetry data anomaly detection model based on bayesian lstm," *Acta Astronautica*, vol. 180, pp. 232–242, 2021.

[23] 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.

[24] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.