Received 17 October 2023, accepted 7 November 2023, date of publication 9 November 2023, date of current version 30 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3331762

RESEARCH ARTICLE

Analysis of Time Series Data Generated From the Internet of Things Using Deep Learning Models

POLYCARP SHIZAWALIYI YAKOI[1], XIANGFU MENG[1], SHUOLIN CUI[1], DANLADI SULEMAN[2], AND XUEYONG YANG[1]
[1]Liaoning Technical University, Xingcheng, Huludao, Liaoning 125105, China
[2]School of Science, Technology and Engineering, University of the Sunshine Coast (UniSC), Sippy Downs, QLD 4556, Australia

Corresponding author: Xiangfu Meng (marxi@126.com)

ABSTRACT The ever-increasing widespread use of the Internet of Things and its applications has generated massive amounts of data. IoT sensor-generated datasets typically have a time-series structure and relational metadata to describe them. Time series data are typically data that have timestamps and can be obtained from sensors and IoT devices. Data prediction is required to maximize the potential of IoT-generated data, and anomaly detection and correction are needed to preserve data quality and integrity. Traditional machine learning models are incapable of analyzing the gigantic amounts of IoT-generated data. On the other hand, deep learning can properly analyze large data volumes, leading to increased use in the IoT domain. This research has examined the use of deep learning models for prediction, anomaly detection, and correction of data generated by IoT devices. The study found that deep learning is widely used in different fields today to analyze IoT-generated data. The research also outlines some challenges being faced while using deep learning models for IoT data analysis. More research is suggested in this study to expose more challenges and tackle the current challenges to achieve better IoT data analysis using deep learning.

INDEX TERMS The Internet of Things (IoT), deep learning, the IoT data analysis, time series.

I. INTRODUCTION

A time series is a collected sequence of repeated observations of a given set of variables over a period. Some examples of series include stock prices, electrocardiograms (ECG), household electricity consumption, and much more.

Definition 1 (Time Series Data): Let $k \in \mathbb{N}, T \subseteq \mathbb{R}+$,

$$\text{then } \{x_k(t) | x_k(t) \in \mathbb{R},\ t \in T\} \tag{1}$$

From (1): assuming k is a natural number and T is a group of timestamps denoted by positive real numbers, then there exists a set (i.e., time series) of values (real numbers denoted by $x_k(t)$) collected at every timestamp.

Definition 2 (Univariate Time Series): a collection of data consisting of single sequential observations that vary over equal time intervals. One example is a monthly household electricity consumption collection in Table 1. If $k = 1$ in (1), then the time series is univariate. Therefore (2) shows the definition of a univariate time series.

$$\{x(t) | x(t) \in \mathbb{R}, t \in T\} \text{ for } T \subseteq \mathbb{R}+ \tag{2}$$

Table 1 shows a snippet of monthly household electricity consumption observations from 1985 to 2017. The data collected (presented in Fig 1) is one example of univariate time series. This series consists of only one data (sales) collected for each timestamp. The data was used to predict electricity consumption in households using time series analysis.

TABLE 1. Monthly household electricity consumption.

Date	Monthly Electricity Consumption
1/1/1985	72.5052
2/1/1985	70.672
3/1/1985	62.4502
4/1/1985	57.4714
5/1/1985	55.3151

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

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

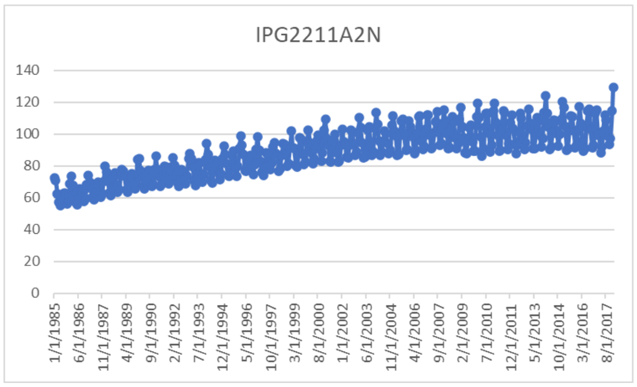

FIGURE 1. Monthly household electricity consumption from 1985 to 2017.

Definition 3 (Multivariate Time Series): By considering (1), we get a multivariate time series by considering values of k greater than 1. Therefore, (3) shows the definition of a multivariate time series.

$$\{x_k(t) | x_k(t) \in \mathbb{R}, t \in T\} \text{ for } k >= 2 \text{ and } T \subseteq \mathbb{R}+ \quad (3)$$

Definition 4 (Multivariate Time Series): a time series consisting of m columns ($A_1, A_2 \ldots A_m$) with n records, where each record refers to the status of the environment of the current timestamp. Table 2 shows a typical multivariate time series. For this series, multiple data are collected for every timestamp. Table 3 shows a real-life multivariate series generated by Uber drivers in a certain city daily. This multivariate series is also shown in Fig 2.

TABLE 2. Typical multivariate time series.

Timestamp	Temperature	A_2	A_1	…	…	A_m
12:05:00	72	0.4	1.1	…	…	21.05

TABLE 3. Data from Uber drivers in a city.

Dispatching base number	Date	Active vehicles	Trips
B02512	1/1/2015	190	1132
B02765	1/1/2015	225	1765
B02764	1/1/2015	3427	29421
B02682	1/1/2015	945	7679

Time series data is sequential data collected at regular intervals during a specific period [1]. IoT sensor-generated datasets typically have a time-series structure and relational metadata to describe them. Time series data such as server metrics, economic indicators, and network data involve data generation at separate times in given periods. Furthermore, predictive models for sales, demand, trends, cycles, and analyzing rapidly fluctuating prices in financial markets use time-series databases [2].

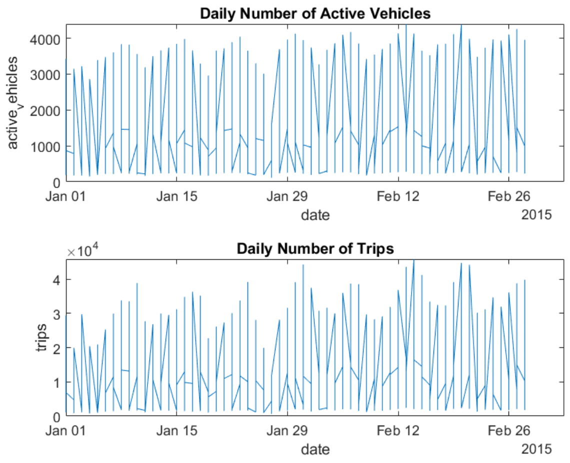

FIGURE 2. Daily number of active vehicles and trips of Uber drivers in a certain city.

Different kinds of time series data, including GPS trajectories and sensor readings, are prone to errors. Existing methods usually concentrate on anomaly detection but not on anomaly repair [3]. For this reason, applications are usually unreliable over the incomplete time series even after using anomaly detection to identify and discard the dirty data. Instead of simply discarding anomalies, it is highly essential to repair them. Time series data analysis has been embraced by different domains in recent times.

The popularity and the need for the implementation of IoT applications are continually growing globally. From healthcare to sports, to education, the list of the various areas of application of IoT is endless. With this comes a generation of an extensive volume of data. As a result, the amount of time-series data generated in different domains has also increased [4]. The need for data quality and information assurance in IoT has only increased, simultaneously with the growth of the use of the technology. For data integrity and usability to be assured, a proper analysis of the data must be performed. Data prediction, anomaly detection, and correction are different forms of analysis that can be performed on IoT data, for different purposes. Deep learning (DL) models have become increasingly popular techniques recently used for the analysis of Internet of Things (IoT) data because they are suitable for processing very complex IoT-generated data [5].

This work focuses on reviewing various works by researchers to perform prediction, anomaly detection, and correction on IoT-generated time series data. The next sections of this study are organized as follows. Section II discusses IoT data, areas of application, and motivation for the analysis of IoT data. Section III introduces deep learning models. Section IV describes the data prediction using deep and other machine learning models presented. Section V reviews the prior efforts on anomaly detection and correction in IoT. Section VI concludes with a summary and outlooks for future works.

II. IoT DATA

IoT data is obtained from sensors and digital and physically connected devices. Typically, IoT devices generate data, which can include ECG, temperature levels, production times, air quality, and water quality, among other things. Because of this, IoT data generation is usually enormous due to the presence of millions of connected devices, and constant generation from small mobile devices. An important part of collected IoT data is the timestamp. When collecting IoT data, the timestamps are usually recorded to ensure reliability and data integrity. IoT data is a kind of time series and has become the main source of time series universally due to the astronomic growth of the usage of IoT applications in recent times.

Data generated from IoT devices usually come in either of these three forms: status, automation, and location data [6]. Status data is raw, unprocessed information that communicates the state of IoT devices. Automation data is produced by automated systems and devices e.g., automatic lights and smart thermostats. Location data conveys the physical location of the system or device. It is frequently employed in manufacturing, warehousing, and logistics. Mohammadi, et al. [7] classified IoT data as big data and streaming data. Streaming data is generated quickly and in small amounts by devices and requires immediate analysis and processing. Big data, on the other hand, refers to massive datasets collected by software and hardware that cannot be stored or analyzed immediately.

A. APPLICATIONS OF IoT DATA

IoT data has numerous applications and benefits, ranging from advantages enjoyed by individuals, industries, and even by governments. For example, in collaboration with IBM, the city of Chicago deployed 300,000 IoT devices that will be used to aid smart grid operations, minimizing energy waste and saving consumers \$170 million [8]. Santander, Spain, which has about 20,000 smart devices installed, has seen another government-level benefit. These IoT devices measure air quality, temperature, humidity, traffic concentration, public transport system conditions and schedules, vehicle velocity, and position, among other intelligent tasks [8]. Just like other fields, the healthcare industry has experienced tremendous advancements through the introduction of IoT. Smart healthcare is facilitated using sensors that obtain data and communicate it via the IoT [9]. IoT in healthcare has improved decision-making and other medical services e.g., identification and monitoring of patients in hospitals, improved patient care procedures, and many more.

Examples of use cases of time series data in IoT include utilities with smart meters that generate billions of data points annually, smart building companies that identify security intrusions in real-time or inefficient energy use, and vision sensors in autonomous vehicles that gather vital information to assist driving. Patient health surveillance, such as in an electrocardiogram, also known as an ECG, which tracks heart activity to determine whether it is functioning appropriately, is another common application of time series data. EarlySense, a remote monitoring device that tracks the vital signs of patients and was created by a specialized nursing facility called Allure in July 2017, is an illustration of patient health surveillance [10].

B. DATA SECURITY AND INTEGRITY

Data security and integrity are critical for ensuring secure and reliable IoT lines of communication. Due to flaws like weak authorization, insufficient software protections, and poor encrypted communication protocols, the majority of IoT devices are typically vulnerable to attacks. These flaws expose the devices to a variety of threats and attacks, raising security and privacy concerns [8]. For this reason, IoT networks should detect anomalies and correct them (swiftly, to avoid any impending harm or attack). IoT devices and sensors generate extreme volumes of data. Also, there are protocol restrictions on message transmission and reception across the various IoT infrastructures and sensor device levels, which makes analyzing IoT data more difficult [11]. DL approaches and other machine learning (ML) methods are needed to effectively analyze the massive amounts of data generated by IoT devices to extract useful information [12]. These tasks are outside the range of the capabilities of traditional techniques used for data analytics. DL approaches present an ideal solution to a range of IoT classification and prediction tasks because they are capable of learning hierarchical representations from the input data. These approaches are potent tools for revealing insight and knowledge concealed in IoT data. They have also improved decision-making in different fields, such as finance, education, healthcare, and security [13]. The IoT can make use of these techniques to better understand underlying patterns in large volumes of data to create the best prediction and recommendation systems.

C. IoT TIME SERIES DATASETS

Datasets are collections of data points curated for training, validating, and testing neural network models. Given the complexities and the vast number of variables in DL models, datasets are typically expansive, often comprising millions of data points and features to ensure comprehensive training to improve accuracy. Depending on the application scenario, datasets can be structured (systematically organized, often in tables or databases) with labeled data points denoting observed occurrences. IoT time series analysis involves working with data collected from IoT devices over time. Any specific dataset chosen for a DL model should align with the research or analysis goals. Table 4 shows a few publicly available datasets that can be used for IoT time series analysis.

III. DEEP LEARNING MODELS

DL is a popular ML approach that has experienced tremendous progress in all traditional ML areas [23], [24], [25]. It is

TABLE 4. Datasets for IoT time series analysis.

Dataset	Description	Number of records	Size (Bytes)	Source
ERA5	Contains hourly observations for a global gridded dataset of over 600000 weather stations for the period 1979 to present; used for climate change prediction.	Depends on the number of weather stations used and the period selected for analysis.	Depends on the resolution and period selected for analysis.	[14]
MIMIC-III	Healthcare, Education, and Research	116 million approximately	24.6 GB (Uncompressed)	[15]
GrowliFlower	Agriculture and Research	5.357 million	14.501 GB	[16]
UCI Machine Learning Repository	Collection of datasets related to air quality monitoring from IoT devices. They include measurements such as pollutant concentrations and meteorological data.	Over 1500	Over 130 GB	[17]
CWRU	Contains vibration sensor data for machine condition monitoring and anomaly detection. Widely used for predictive maintenance applications.	Over 200, 000	Over 12 GB	[18]
CityPulse	This project provides datasets related to environmental monitoring from various European cities. This includes data on air quality, noise levels, and other environmental factors.	2.375 million	1.3 GB	[19]
Pecan Street Project	Provides access to smart grid data, which includes electricity consumption data from residential households, along with environmental and weather data	Over 735, 961	0.419 GB	[20]
Seismic	USGS archived earthquake data.	Over 2.5 million	Over 500 GB	[21]
MetroPT dataset	A collection of observed pressure of air, Temperature, and ampere for predictive maintenance.	10.99 million	Over 1.1 GB	[22]

(one of) the best tool(s) used today for the learning process and analytics of IoT data, as well as playing an important role in making IoT smarter [5], [12], [26]. It is a sophisticated ML technique that enables predictive learning in the IoT domain. It is a process that extracts data in the same way that neural networks do in a hierarchical learning form. It transforms data into abstract representations, which aids in the learning of features [26], [27]. Different DL techniques are widely utilized in image recognition and time-series inference for IoT applications. These neural networks, initially inspired by the human brain, try to imitate how the brain of a person works, however, they are incapable of matching its impressive ability to "learn" from huge volumes of data [28].

DL techniques, just like any other machine learning algorithm, can also be trained. They can be classified as supervised, unsupervised, and semi-supervised learning. A typical deep learning model has multiple layers (usually three or more), each of which builds on the one before it to improve the predictive model or classification. This sequence of computations in DL networks is called forward propagation. The input and output layers of the deep learning network are called visible layers. The DL model accepts and processes data in the input layer, with the output layer performing the final operation – either prediction or classification. Another method called back propagation utilizes algorithms like gradient descent to determine prediction errors before changing the function's weights and biases by going back through all the layers iteratively to train the model. A classic deep learning model makes predictions and necessary corrections for any errors by using both forward and back propagation simultaneously. The algorithm gradually enhances its accuracy as time goes on [28]. Generally, DL models can be very complex, and different neural networks are used for specific problems or datasets. In recent times recurrent neural networks and LSTMs have become the most popular DL models that are used for IoT data analysis. The transformer model, proposed in 2017 has gradually become a very important model also.

A. RECURRENT NEURAL NETWORKS (RNN)

These are neural networks in which the outputs of every stage are used as inputs for the next. In traditional neural networks, each step progresses without any input from the previous one, meaning memory is not preserved. In other words, RNNs are networks comprised of loops to enable them to reuse information. Traditional feedforward neural networks process a fixed amount of input data at the same time and produce a predetermined number of outputs. RNNs, on the other hand, do not process all the input immediately. Instead, they repeatedly separate them into a sequence. RNNs perform a sequence of computations for each step before providing an output. The output (known as the hidden state) is subsequently combined with the next input in the sequence to generate another output. This method is repeated until when the model is programmed to stop, or the input sequence is complete. The ability to utilize important data from previous steps is essential for RNNs to solve sequential problems successfully.

RNNs are compared by Saxena [29] to a person watching a video and remembering the previous scene or a person reading a book who remembers what happened in the previous chapter. Similarly, they recall earlier information and apply it to the current input. They consist of loops that allow information to be retained. One of the main advantages of RNNs is their ability to connect between earlier information and the current task, for instance, by utilizing previous video frames to understand the current frame. RNNs would be very helpful if they could accomplish this. Can they, though? Perhaps, though not always. Due to the vanishing gradient problem that arises while working with relatively long data sequences, RNNs are usually unable to remember long-term dependencies. This is not a problem with LSTMs, they avoid problems with long-term dependency (more on LSTM in Section III-B).

Fig 3 shows that RNNs contain neural networks (say A) that take an input (say x_t), process it, and provide an output (say h_t). A is a simple one-layered module for processing the input. In the next step of the network, A uses the previous value of h_t for processing.

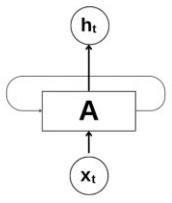

FIGURE 3. Loops in RNNs.

An RNN is a chain of multiple duplicates of the same network, each handing a message to the next one. Fig 4 shows an RNN in its unrolled form. In classic RNNs, the repeating module is usually made up of a simple structure, e.g., a single tanh layer. The tanh function serves as an activation function, which ensures the value of the latest information is between −1 and 1.

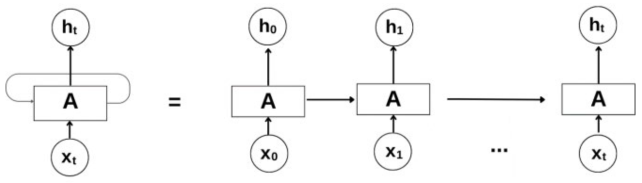

FIGURE 4. Unrolled form of RNNs.

The long-term dependency problem in RNNs becomes evident when the value of t in Fig 4 becomes extremely large. This means that as the value of t increases, the ability of RNNs to remember diminishes. Fig 5 shows the repeating module of a classic RNN with only one neural network layer.

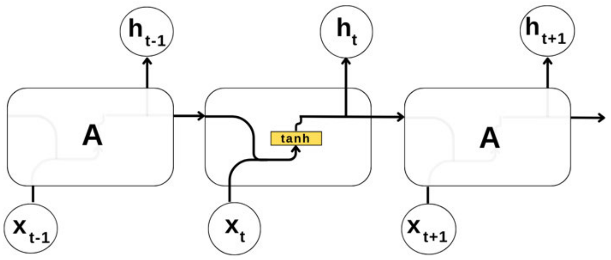

FIGURE 5. The repeating module in classic RNNs.

B. LONG-SHORT-TERM MEMORY (LSTM)

LSTM networks are RNNs usually used to address a long-standing problem in latent variable models: long-term information preservation and short-term input skipping. Along with other DL algorithms, for example, RNNs, the LSTM is used for analysis and to detect anomalies in time series (i.e., data associated with timestamps). It compares an existing series with the input series to detect anomalies. As seen in [30], LSTM networks can also be used for prediction (forecasting) in time series data generated from IoT. They can learn order dependence in time series prediction. This is a necessary behavior in many domains that involve challenging predictions and complex problems, including machine translation, speech recognition, and many others. LSTM is equipped to deal with long-term dependencies [29]. As shown by Google Scholar, LSTM was cited over 16,000 times in only one year, in 2021 [31]. This demonstrates LSTM applications in a variety of areas, which include healthcare, time series prediction, robotics, chatbots, etc. It is also the most cited neural network of the twentieth century [32].

After getting a clear picture of what RNNs are (in the previous section), we look at LSTM (and what they have that makes them different). The LSTM is a chain-like network as well, however unlike RNNs, it uses more complex neural networks ("A"). The read-write-and-forget concept governs how LSTMs operate. Given an input of information, the network only reads and writes the data that will be useful for predicting the output. The A in LSTMs has four layers as opposed to the single-layered A in RNNs. Each of these four layers has a unique interaction with the others; therefore, the network transfers the selected information only.

The four layers are one single neural network layer which is usually a simple module e.g., the tanh layer (the same as the one in RNN), and 3 multiplicative gates (forget, input and output gates) [29]. The forget gate processes information from the previous timestamp and then decide to remember relevant information and forget irrelevant ones. The input gate is used to process the input from the current timestamp and then add to or update the current information. The output gate is then used to pass updated data from the current timestamp to the next. Fig 6 shows the repeating module in LSTMs containing 4 (i.e. 3 σs and 1 tanh) layers of neural networks. In recent times, different researchers use LSTM networks in different forms for detecting and correcting anomalies in time series data (more in Section V); this trend will continue as the LSTMs continue gaining more popularity.

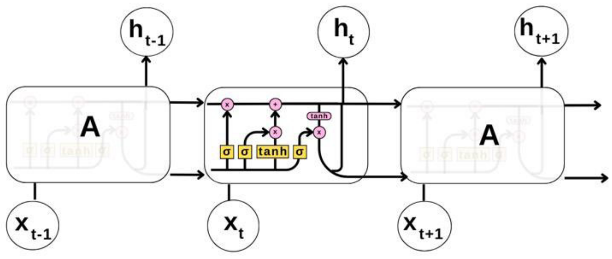

FIGURE 6. The repeating module in LSTMs.

C. TRANSFORMER MODELS

RNNs, LSTMs, and other variations of the RNN (such as gated RNNs) are revolutionary approaches used for sequence modeling. However, these solutions are usually slow and ineffective for problems that require sequence-to-sequence solutions. The Transformer model introduced in Vaswani, et al. [33] is a neural network that offers both speed and effectiveness through parallel computation and efficient long-range dependency modeling. It revolutionized the field of natural language processing by leveraging self-attention mechanisms to capture dependencies between elements in a sequence.

A transformer is a DL model that utilizes the self-attention mechanism and distinctly weights the importance of every element of the input data. The transformer model observes relationships in sequential data, e.g., the words in this sentence, to learn context and then decipher meaning. It was developed initially for machine translation, but because of how well it performed, it was quickly applied to other fields like music, image creation, audio production, and text summarization [34]. Currently, the transformer is used with considerable success for time series data analysis [35].

The Transformer is made up of an encoder and a decoder, shown in Fig 7. The encoder processes the input sequence, and the decoder generates the output sequence. Every encoder and decoder layer in the model contains self-attention and feed-forward neural network sublayers (see Fig 8). The self-attention sublayer captures dependencies within the sequence, and the feed-forward sublayer applies non-linear transformations to the representations.

Self-Attention mechanism is the main component of the Transformer model. The attention function maps 3 vectors (a query and a collection of key-value pairs) to an output that is itself a vector. With this function, the model can weigh the importance of different variables in the input sequence while producing an output. Based on their similarities to the query and key vectors, self-attention calculates a weighted sum of values at different places in the sequence. This mechanism enables the model to attend to different parts of the sequence

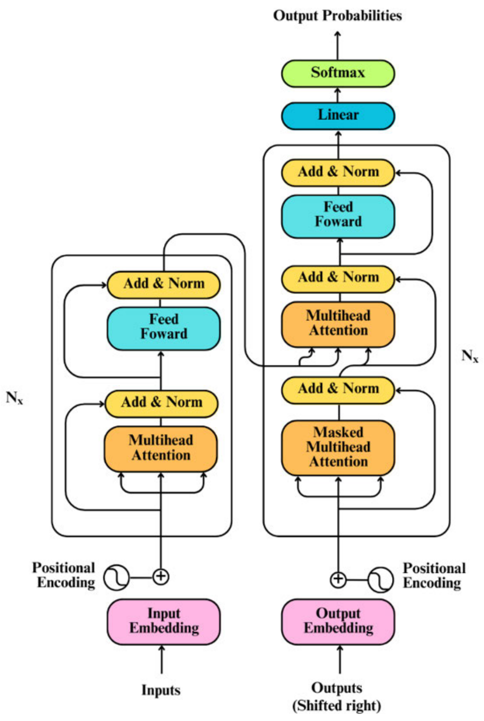

FIGURE 7. Transformer architecture.

simultaneously, capturing global dependencies. Equation (4) shows the matrix of outputs (where Q is Query, and K and V are the key and value vectors).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{4}$$

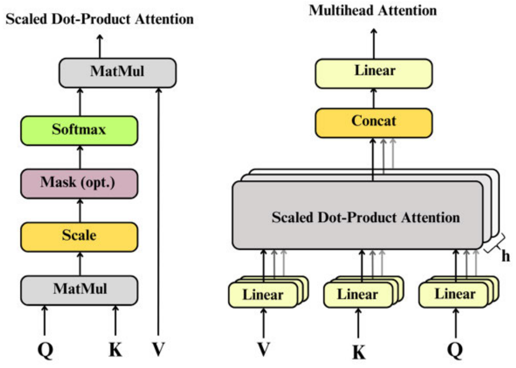

FIGURE 8. Scaled dot-product attention (left). Multi-head attention (right).

Multi-Head Attention (MHA), shown in Fig 8 is a key component of the Transformer model. MHA allows the model to evaluate distinct aspects/dimensions of the input sequence at separate times/heads, thereby enabling it to better encode and decode the input information.

MHA works by dividing the input sequence into multiple sub-sequences (or 'heads'), and computing attention scores for each head independently and concurrently. Every head computes a set of attention scores that capture the importance of different parts of the input sequence for that head. These attention scores are then concatenated across heads, resulting in a single set of attention scores that capture the importance of parts of the input sequence for all heads.

Finally, these attention scores are used to compute a weighted sum of the input sequence and then passed through a feedforward neural network to generate the output sequence. The use of multiple heads allows the Transformer to encode richer contextual information into its representations, leading to improved performance on language understanding tasks such as machine translation, language generation, and question answering.

Embeddings are a form of representation learning, in which each unique word or token in the vocabulary is mapped to a unique vector in a low-dimensional space. They are learned during the training process and allow the model to encode the meaning and context of each word or token in a fixed-length vector. The embeddings are trained to capture the relationships between words, such as synonyms (similar meaning) and antonyms (opposite meaning), as well as the structure of the language.

Softmax is a type of activation function that is commonly used in neural networks for classification tasks. It is added to the output of the embedding layer to convert the embeddings into probabilities that represent the likelihood of each token being selected as the next word in the output sequence. Softmax allows the model to produce a distribution over the vocabulary tokens that sums to 1 (one), representing the probability of each token being selected.

In the Transformer model, the output of the embedding layer is typically passed into a feedforward neural network; a softmax layer follows afterward to generate the distribution over the tokens. The distribution is then used to select the next token in the output sequence by sampling from this distribution with a random number generator. This process is repeated until the output sequence reaches a certain length or a special end-of-sequence token is generated.

By combining embeddings and softmax, a Transformer can produce high-quality output sequences that capture the meaning and syntax of the input sequence while generating relevant and grammatically correct output.

Being a self-attention-based model, the transformer does not contain any convolution or recurrence, like the other popular neural networks; meaning it relies on its ability to handle parts of the input sequence by considering their relevance to the current output sequence. However, this poses a challenge when trying to capture the order and the corresponding positions of the tokens in the sequence, as self-attention alone cannot capture this information.

To address this issue, positional encoding is added to the input sequence as an additional signal that captures the order and position of each token in the sequence. *Positional encoding* is a critical component of the Transformer model that allows it to encode the order and position of words or tokens in a sequence. This encoding is achieved by first mapping each token in the sequence to a unique integer ID, and then using these IDs to compute sine and cosine functions that produce fixed-length embeddings for each token in the sequence. These embeddings, called *positional embeddings*, are added to the word embeddings of each token in the sequence to capture their position in the sequence. The resulting combined embeddings are then fed into the Transformer encoder, which uses self-attention and feedforward neural networks to encode the input sequence and generate output tokens.

By adding positional encoding to the input sequence, the Transformer can capture not only the meaning of each token but also its position within the sequence, enabling it to generate high-quality output sequences that respect the original order and distance relationships between tokens in the input. Positional encoding enables the transformer to take a complete sequence (say a sentence in English) at once and analyze its tokens simultaneously. For this reason, the transformer is faster than convolutional and recurrent neural networks.

During training, the Transformer model is optimized using supervised learning. The model minimizes a loss function, such as the cross-entropy loss, to predict the correct output sequence of a given input sequence. The model parameters and attention weights are updated through backpropagation and gradient descent.

D. PERFORMANCE METRICS FOR DL MODELS

To find out which DL model is most suitable for specific IoT scenarios, it is essential to evaluate the performance metrics of those models. To compute any performance metrics, it is important to know about the *confusion matrix*. The confusion matrix is a combination of scenarios that is used to describe a binary classification problem, such as how to judge whether someone has cancer or other diseases. Nevertheless, every kind of model will always make wrong judgments. So, the data that is originally correct will be defined as true and false samples; the data that has been classified will be defined as positive and negative samples. Therefore, there are four kinds of samples in the confusion matrix - *TP* (True Positive), *TN* (True Negative), *FP* (False Positive,) and *FN* (False Negative). It is usually straightforward to know that only the *TP* and *TN* are accurate samples. So, to learn about the probability that the sample will be classified correctly, the *accuracy* is defined in (5).

$$\text{accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \tag{5}$$

The accuracy will show the proportion of all samples that predicted correctly to all test samples. It should be noted that for the model, both *TP* and *FN* are originally the true samples, meaning the *TP* accounts for the true data.

$$\text{recall} = \frac{(\text{TP})}{(\text{TP} + \text{FN})} \tag{6}$$

The *recall,* which is the proportion of all positive cases that were correctly predicted by the model is computed using (6).

Now that the *TP* and *FP* are recognized as the positive samples, the *precision* computed in (7) is used to learn the percentage of the *TP* makes up the positive samples.

$$\text{precision} = \frac{(\text{TP})}{(\text{TP} + \text{FP})} \tag{7}$$

After computing the precision and the recall, the *F1 score* shown in (8) which is the harmonic mean will be defined as well. The F1 score combines the recall and accuracy to determine the accuracy of a model. A higher F1 score shows that the model is having a better performance.

$$\text{F1} = \frac{(2\text{TP})}{(2\text{TP} + \text{FP} + \text{FN})} \tag{8}$$

Despite defining the accuracy, possibilities exist in some situations, that it may not distinguish the originally correct samples and the predicted correct samples; this inhibits the model's ability to get the right data. So, to solve this problem, *TPR* (True Positive Rate) and *FPR* (False Positive Rate) are computed in (9) and (10) respectively.

$$\text{TPR} = \frac{(\text{TP})}{(\text{TP} + \text{FN})} \tag{9}$$

For *TPR*, it represents the proportion of all true instances in the positive class predicted by the classifier. Numerically, the TPR is the same as the recall. *FPR* represents the proportion of all false instances in the positive class predicted by the classifier.

$$\text{FPR} = \frac{(\text{FP})}{(\text{FP} + \text{FN})} \tag{10}$$

Suppose that the probability of a series of samples being divided into positive classes has been obtained, and then sorted by size; the next action is to take the "*Score*" value as the threshold in turn from high to low. When the probability that the test sample belongs to a positive sample is greater than or equal to this threshold, it is considered a positive sample, otherwise, it is a negative sample. Each time a different threshold is picked, a set of *FPR* and *TPR* can be obtained, that is, a point on the *ROC* curve. So, the *ROC* curve can be plotted with multiple values. The so-called *ROC* refers to *receiver operating characteristics*. Each point on the *ROC* curve reflects susceptibility to the same signaling stimulus. The *area under the curve*, *AUC*, can tell whether the model works well numerically. When the AUC is higher the model performs better.

Usually, more performance metrics can be utilized to measure the performance of DL models. The most important point to consider when choosing any metric is that it aligns with the goals of the DL model, the characteristics of the dataset, as well as the impact of outliers on the analysis. Additionally, multiple metrics can be used to get a more comprehensive view of the performance of the model.

IV. CLASSICAL DL MODELS FOR DATA PREDICTION

Data prediction is extremely important for IoT applications, but it is impossible to make accurate predictions due to the enormous and high-dimension data generated, and inevitable missing values. The predictive analysis capability of DL and other machine learning algorithms can be leveraged to address a variety of issues, including prediction problems in IoT [5], [36], [37].

To address the prediction challenge, an integrated online prediction model that can solve future value prediction and missing value imputation in low-dimensional space simultaneously has been proposed [38]. This approach integrates classic prediction models into the objective function of matrix factorization before employing a rolling prediction mechanism to implement a very efficient data prediction for the IoT.

One classic DL model used for data prediction is the *TS-TCC* presented in Eldele et al. [39], which is used to learn representation of unlabeled time series data. TS-TCC uses weak and strong augmentations to split raw time series data into 2 distinct but correlated views. Contrastive approaches attempt to maximize the similarity between distinct views of the same sample while reducing its similarity to other samples. As a result, it is critical to create appropriate data augmentations for contrastive learning. After the transformation, the model creates a complex cross-view prediction job and employs a temporal contrasting module to learn reliable temporal representations. To further develop discriminative representations, the TS-TCC employs a contextual contrasting module based on the contexts from the temporal contrasting module. The TS-TCC algorithm aims to minimize the similarity between contexts of different samples while maximizing the similarity between contexts of the same sample. Fig 9 shows the architecture of the TS-TCC.

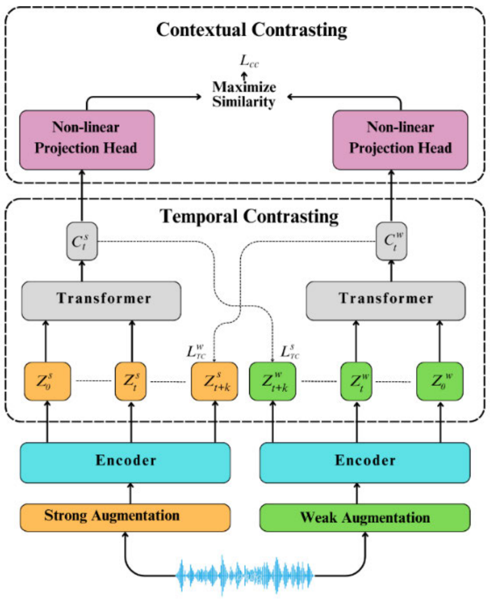

FIGURE 9. The architecture of the TS-TCC model.

As already stated, IoT devices generate massive data, making the accuracy of prediction of real-time spatial information very cumbersome. Inagaki et al. [40] proposed an IoT system that minimizes the transmitted data utilized as input for prediction in real-time, sustaining the prediction accuracy by using a machine learning-based system. The system - which is made up of mobile nodes and an edge server receives collected data, aggregates and preprocesses the data, and then uses it to predict the next spatial information in real-time by using feature selection. A similar approach is presented by [41] for analyzing data obtained by numerous IoT devices for predicting health status in real-time. Metaheuristic algorithms have been employed by different researchers for feature extraction, which is an important process in the prediction of data [42].

Eraliev and Lee [37] also present the use of LSTMs to perform climate prediction by analyzing time series data obtained directly from an indoor greenhouse used for hydroponics. The input features are run through several LSTM layers by the LSTM model. The predicted levels of humidity, temperature, and CO_2 are produced by the output layer. According to Fig 10a, the LSTM structure is built by multivariate time series and routing them via the LSTM layer, where the outputs and neurons are of the same number. The LSTM layer has 64 cells in it and Fig 10b shows the configuration of one single cell in the layer.

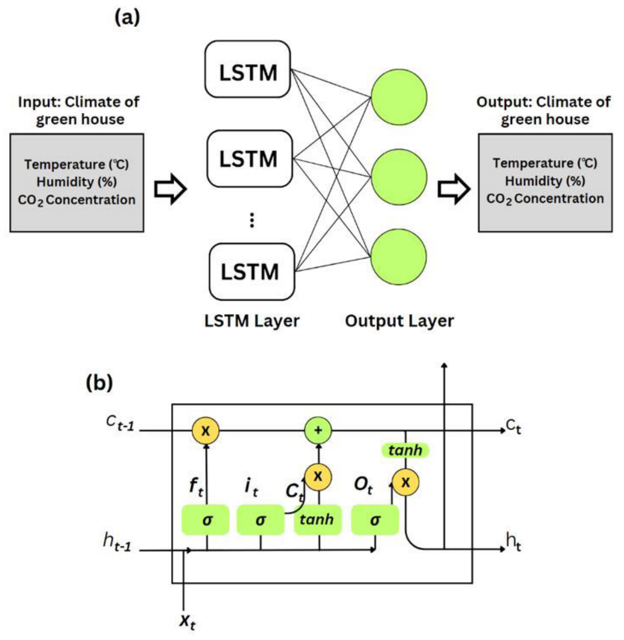

FIGURE 10. **(a) Structure of LSTM used in [28]. (b) The structure of a single cell of the LSTM.**

The LSTM network is made up of cells that predict the next output based on prior state input and current input. The cells oversee evaluating the significance of the input, keeping it inside the memory, and either refusing to acknowledge it or forwarding it onto the next loop, enabling the RNN to address the vanishing gradient usually associated with RNNs. In addition, Eraliev and Lee [37] also present two other DL-based prediction methods using a deep neural network and a 1-D CNN model. Just like the LSTM model initially discussed in the article, these two are used to predict climatic conditions at various times in greenhouses.

Cao et al. [43] present a DL model for multi-task learning to be used in time series prediction. This model considers many parameters and combines different features. After extracting and fusing these features, the model forecasts time series using LSTM layers. From the model architecture, shown in Fig 11, the model takes in three inputs - V (Maximum Connections value), M (current moment), and R (Maximum Connections change rate). Afterwards, feature extraction is performed.

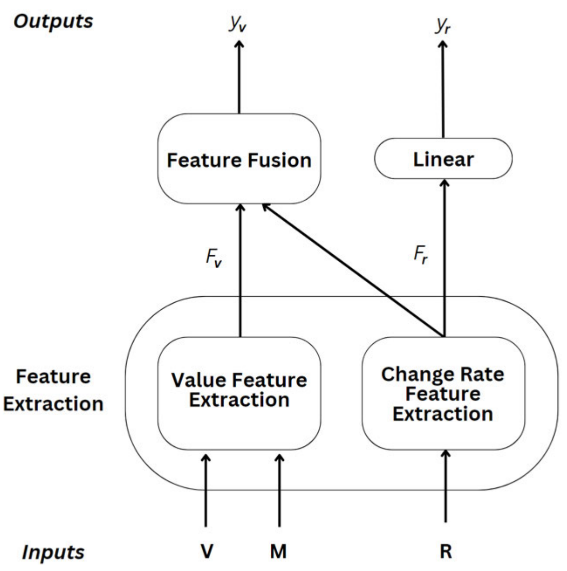

FIGURE 11. **The architecture of the deep multi-task learning model.**

Both feature extraction segments have two sections: feature transform and LSTM layers. The value feature extraction algorithm converts the inputs M and V into the predicted value of the feature F_v. Change rate feature extraction converts the change rate R of Maximum Connections to a feature matrix of the predicted change rate - F_r. Following that, F_v and F_r from the extraction sections were fused in the next stage to produce the output sequence $y_v = y_{v1}, y_{v2} \ldots y_{vN}$. Meanwhile, as a related activity, the change rate characteristic is employed to optimize anticipated maximum connections. The model yields $y_r = y_{r1}, y_{r2}, \ldots, y_{rN}$ by calculating the loss in the change rate, which is utilized in conjunction with value loss in the complete training model.

Furthermore, it employs multi-task learning to jointly maximize its capabilities. It also takes the predicted change rate of maximum connections (related task) into account as an inductive bias when tuning the predicted maximum connections (primary task). The results of the experiments suggest that the change rate helps to increase prediction accuracy.

In Liu et al. [44], using data mining techniques, a machine learning model was presented, to forecast substation project costs and enhance power grid companies' capacity for cost prediction. This model, when properly implemented, has the potential to provide greater insight for decisions related to cost management. Tkachenko et al. [45] developed a new solution based on neural network tools to increase the precision of completing prediction tasks for recovering missing IoT data. The proposed method consists of a neural-like successive geometric transformation model structure and two successive general regression neural networks. This method aids in resolving issues regarding missing values from real-time data obtained by IoT devices from monitoring air in the environment. In [1], a time series analysis and prediction on IoT data obtained from three locations was performed to predict future air quality by using the machine learning algorithm - linear regression. For predictive analytics of dynamic and complex IoT data streams, Akbar et al. [46] developed an adaptive algorithm for prediction named adaptive moving window regression. In addition to IoT, the system can accurately predict complex events in several other fields with a 96 percent success rate. Chahal and Gulia [26] ascertain firmly in their study that deep learning is the best tool for the predictive analysis of IoT data. Classical DL models for prediction in time series data generated by IoT are presented in Table 5.

V. CLASSICAL DL MODELS FOR ANOMALY DETECTION AND CORRECTION

Anomalies are (unusual) observations that differ from (or do not follow the regular pattern of) other observations. These anomalies can significantly degrade the quality of data [55]. Recently, various techniques have been invented for detecting anomalies in IoT and other fields. This is an extremely crucial step to curb the growing menace of data integrity and security issues. Due to the likelihood of noise and the lack of labels in the sensor readings, anomaly detection in IoT sensor data has grown in importance. For this reason, many scientists and researchers have used machine learning-based techniques for intrusion and anomaly detection [56]. However, there is a high degree of correlation between the sensor data points, making it extremely difficult for traditional machine-learning techniques to detect anomalies [57]. Furthermore, the inability of typical statistical and machine learning algorithms to detect anomalies is due to the size and speed of the data collected by the IoT sensors.

Ji et al. [58] present LSTMAD - an anomaly detection framework based on LSTM which detects anomalies in univariate time series data. This framework learns the structure of normal (non-anomalous) training data and then proceeds to detect anomalies by applying a statistical technique using the predicted error for the data that was observed. Fig 12(a) shows the architecture of the LSTMAD framework which consists of four modules. Fig 12(b) shows the LSTM model of the framework, which consists of five layers.

TABLE 5. Classical deep learning models used for prediction in IoT time series data.

Name	DL/ML used	Source	Application
CNN-RNN Framework for Crop Yield Prediction	CNN, RNN	[47]	Crop yield prediction
TS-TCC	Transformer	[39]	Fault Diagnosis, epilepsy Seizure Prediction, etc.
RNNs for Multivariate Time Series with Missing Values	GRU	[48]	Predictive tasks in health applications
Transformer Self-Attention for Time Series Forecasting	Transformer	[34]	Time series forecasting
Predicting Infectious Disease Using DL & Big Data	LSTM	[49]	Predicting Infectious Disease
TSA-TNTM	Transformer	[38]	Identification of threats in cyber threat intelligence.
ABCDM	CNN, RNN	[50]	Sentiment polarity detection
LSTM and BiLSTM in Forecasting Time Series	LSTM, Bidirectional LSTM	[51]	Forecasting in time series analysis
Hybrid CNN-LSTM	CNN, LSTM	[52]	Short-Term Individual Household Load Forecasting
Residual Convolutional LSTM for Tweet Count Prediction	Convolutional LSTM	[53]	Tweet Count Prediction
RCLSTM Model	LSTM	[54]	Traffic prediction and user-mobility forecasting

The *noise reduction* module is used to process the input time series data to clean any noise signal that may affect the accuracy of the results of the computation. The noise reduction layer uses algorithms like the S-G filter (which is used to smooth digital data to increase the precision while preserving the data's original properties. In the *normalization* module, the series is normalized to follow a 0-1 normal distribution. The LSTM module is composed of five layers, the first of which is the input layer, which contains L1 nodes, suggesting that a subseries with L1 elements was utilized as input to a fully linked hidden layer. To analyze the data that comes from the input layer, three hidden layers (consisting of LSTM units shown in Fig 12b), are used. In the output layer, there is only one node - Y. The LSMT module predicts the next values using inference from the historical data. After training the LSTM module, a search for anomalies is performed in the *anomaly detection* module.

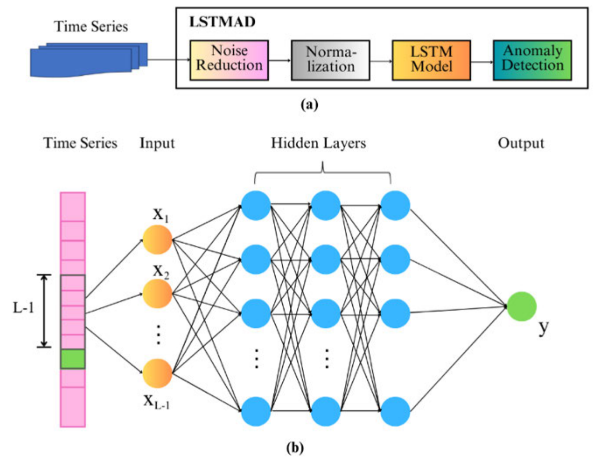

FIGURE 12. Flowchart of the LSTMAD framework.

LSTMAD does not require previous information; it can learn the context of the sequence data from an ordinary signal and then identify abnormal components by using the prediction error for the observed data. It is also not affected by the length of the sliding window, making it a scalable technique for use in different applications. It is also useful for predicting anomalies in real time, particularly when the fundamental physical process has not been entirely understood and defined.

LSTM-Gauss-NBayes proposed by Xie et al. [35] is another classic DL model used to detect anomalies in IoT time series. Initially, the LSTM-Gauss-NBayes (shown in Fig 13) uses downsampling to simplify recognizing patterns in the original time series by reducing the dimensions. The model then normalizes the data by performing a linear transformation using the min-max normalization usually used to analyze time series data. This speed up the rate of convergence of the model. The LSTM-Gauss-NBayes prediction model is constructed using the normal time series training data. The estimated error from the prediction model is then used to build a Gaussian naive Bayes model to determine whether the initial data set is anomalous.

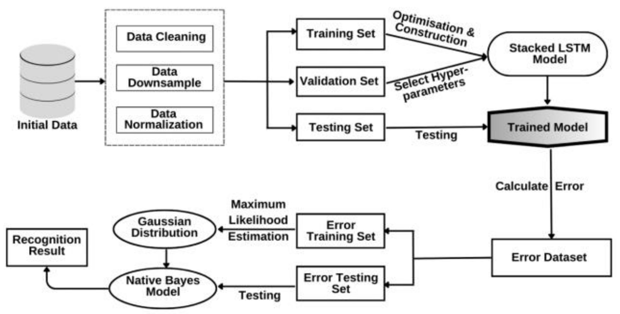

FIGURE 13. Construction procedure of LSTM-Gauss-NBayes.

Ullah and Mahmoud [36] introduced a unique intrusion detection solution for detecting anomalies in IoT. Initially, a multiclass classification model is built using a CNN model. The suggested model is then implemented using 1D, 2D, and 3D CNNs. A CNN multiclass pre-trained model is used to achieve binary and multiclass classification via transfer learning. Considering accuracy, precision, F1 score, and recall, the model performed exceptionally in testing. Zhang et al. [3] intelligently combined the temporal nature of anomaly identification with the popular minimal change principle in data repairing to provide a technique of iteratively correcting abnormalities in data obtained from time series. Table 6 presents DL models that have been used in recent times for anomaly detection and repair in IoT time series data.

VI. CHALLENGES OF USING DEEP LEARNING FOR IoT DATA ANALYSIS

Using DL for IoT data analysis has the ability to extract useful information and make predictions in a variety of applications. However, challenges remain that indicate there is still an urgent need for improvement despite the advances made by DL methodologies for evaluating IoT time series data in recent times. Fig 14 shows an overview of the challenges discussed in this section.

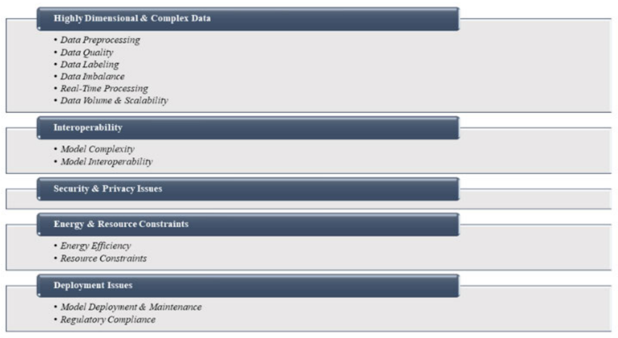

FIGURE 14. Challenges facing IoT data analysis using DL.

A. HIGHLY DIMENSIONAL AND COMPLEX DATA

One of the most fundamental challenges in using DL to analyze IoT data is the data itself. Time series data generated from IoT is highly dynamic, with high dimensionality and complexity; and this is a serious challenge that should be addressed immediately. In addition, the absence of sufficient data sets for adequate model training is a serious concern that must be addressed urgently. Furthermore, data generated by IoT devices is not always in an appropriate shape to be fed into DL models; thus, preprocessing is critical before training the models. Preprocessing is more complicated in IoT applications since the system deals with data from several sources that may have varied forms and distributions while reporting missing data. The practical use of data collection systems is a critical research issue.

DL models usually require clean training datasets that are not polluted by anomalies for learning the "normal

TABLE 6. Classical deep learning models used for anomaly detection and repair in IoT time series data.

Name	Source	DL Model(s) Used	Use-cases
LSTM-Gauss-NBayes	[59]	LSTM, Bayes model, and Gaussian distribution	General time series anomaly detection
LSTMAD framework	[58]	LSTM	Real-time anomaly detection
Anomaly-based intrusion detection model for IoT networks	[60]	CNN	Intrusion detection in IoT networks
OmniAnomaly	[61]	Stochastic RNN	multivariate time series anomaly detection
Anomaly detection in aircraft data using RNN	[62]	RNN	Anomaly detection in aircraft data
VAE-LSTM hybrid model	[63]	LSTM	Identifying anomalies in time series
LSTM-Based Time-Series Anomaly Detection	[64]	LSTM	Anomaly Detection in Rail Transit Operation Environments
IoT multivariate time series anomaly detection	[65]	Adversarial transformer	Multivariate time series anomaly detection
Time-series anomaly detection with stacked Transformer representations and 1D convolutional network	[66]	Transformer	Time-series anomaly detection
Anomaly Transformer	[67]	Transformer	Anomaly detection
A model-agnostic sample filtering method	[68]	LSTMAE	Anomaly detection on time series with contaminated training data
DCT-GAN	[69]	Transformer	Time series anomaly detection
BTAD	[70]	Bi-Transformer	Anomaly detection of multivariate time series
TGAN-AD	[71]	Transformer	Anomaly detection of multivariate time series

profiles” of time series data. This is extremely difficult, if not impossible because clean datasets are rarely available in practice [68]. Different applications built on dirty time series data are completely unreliable. Filtering out dirty data is frequently done via anomaly detection over time series [3]. In other words, observed anomalous data points are simply ignored as useless noise. With a high number of consecutive data points removed, the applications are usually run on a relatively partial time series. As a result, correcting dirty values (or abnormalities) in time series data could enhance applications. Furthermore, the amount of noise in datasets can vary significantly, and noisy occurrences are frequently dispersed randomly. For this reason, models are susceptible to noise in the input data, and their performance is significantly affected. A repair that is relatively close to the truth benefits the applications immensely - attaining this is a significant task.

In addition to having high noise characteristics, the handling of heterogeneous data supplied by IoT devices is a very significant challenge. IoT devices generate vast amounts of data of varying types and scales, for example, signal frequency and network traffic, which, while originating from the same source, will have distinct forms. Data of the same type might also have different scales, for example packets and bytes all belong to network features but they use different scales. If data mining is performed directly on these original time series data, it will not only use a large amount of storage and computing time, but it will also have an impact on the accuracy and reliability of the algorithm. Analyzing the massive amounts of data generated by IoT devices and networks can be time-consuming and difficult to manage appropriately [72]. Also, as the length of the time series increases, learning normal patterns of time series and identifying anomalies becomes more complicated. Additionally, given the need to optimize massive weight parameters in neural networks, DL techniques usually work while assuming that data is adequate and balanced [27]. However, this does not always work for problems in specific scenarios. In many circumstances, the volume of datasets is limited by complex and expensive data-collecting techniques. Furthermore, such processes frequently exhibit extremely unequal class distributions, with instances from one class significantly outnumbering instances from other classes. For example, in clinical cases, there is invariably less data from therapy groups than from the normal groups [73]. Recent years have seen a significant amount of research on the huge size and intricate patterns of time series, enabling scholars to create specialized deep-learning models for identifying abnormal patterns [74]. Even more, a few assessment measures have also been utilized to clearly demonstrate how restricted and imbalanced data can impair deep learning performance.

Owing to its high dimensionality and complexity, time series tend to be uncertain at any one time or change frequently in many cases. Because the time series cannot be predicted in advance in these instances, prediction-based anomaly identification is rendered ineffective. This inhibits the capacity of prediction models to make predictions for the long term [74]. This also inhibits the ability of a DL method to detect anomalies on demand, which is a very essential requirement for applications that require real-time processing. If a system takes more time to process observations than the time to make estimations, then in the long run, the computational resources provided to the system will be exhausted, causing a system failure [75]. Due to the high speed of data generation by IoT devices, like in cases of medical emergencies, it is difficult to process and analyze obtained data in real-time to perform actions on demand. For this reason, real-time data processing capabilities should be added to DL models to ensure an effective and scalable solution. By addressing this problem, DL algorithms will be able to recognize anomalies in real-time.

The absence of labeled anomalies is another major challenge that has consistently impacted IoT time series analysis [75]. Failure modes are insufficient for use as labeled training data because they are extremely rare in most industrial settings. The shortage of failure modes makes gathering sufficient labeled training data resource- and time-intensive. Regardless, when labeled data are acquired, irregularity among normal and abnormal data impedes the training of the model. Without labeled anomalies, unsupervised or semi-supervised approaches are required. This causes a huge number of normal instances to be mistakenly recognized as anomalies. Similarly, missing values are common in multivariate time series data in practical applications such as health care, geoscience, and biology. Data prediction is critical for IoT applications, but it is challenging to create reliable predictions due to the massive amounts of data collected and the unavoidable missing values. Missing values and patterns have been found to be frequently connected with target labels in time series prediction and similar tasks; this is called informative missingness. Although some work has been done to address this issue, there has been relatively little work on utilizing missing patterns for effective imputation and enhancing prediction performance, as well as missing IoT data recovery. Therefore, one of the key challenges is finding a mechanism to minimize false positives and enhance recall rates of detection. This scenario is viewed as a considerable cost related to failure in anomaly detection.

B. PRIVACY AND SECURITY ISSUES

Recent obstacles that diminish the applicability of DL to evaluate IoT data are privacy and security concerns; IoT data are usually obtained from multiple sources, the quality, integrity, and assurance of which cannot be guaranteed. Maintaining data privacy and secrecy is a big challenge in many IoT applications, as IoT massive data is transmitted for inspection via the Internet, making it available to unauthorized parties. This means that there is a possibility that the DL is learning from training data that has already been tampered with. For this reason, sometimes some useful data might be withheld by appropriate authorities, for security reasons. For example, clinical information is not usually made available for the public due to privacy concerns and ethical considerations, resulting in an even greater imbalance in available data. In this case, DL models must be updated using specific approaches for detecting irregular or erroneous input.

C. INTEROPERABILITY

DL models can be very complex, making them difficult to interpret and explain. In scenarios where interpretability is crucial (e.g., healthcare or finance), this can be a significant challenge. The need to improve interpretability has increased in recent years. A level of interpretability is necessary in situations when anomaly detection is being used as an instrument for diagnosis. Nevertheless, the majority of anomaly detection research ignores the problem of interpreting anomalies and instead focuses solely on detecting precision. To show the importance of this particular challenge, one workshop on Explainable Artificial Intelligence (XAI) from 2017 was specifically arranged by IJCAI. In addition, a study on comprehending black-box predictions was given the title of best paper at ICML 2017 [76].

Furthermore, anomaly identification in multivariate time series data is particularly difficult because it depends on the simultaneous analysis of temporal dependencies and variable relationships [74] [77]. Anomaly detectors need contextual information, including temporal, environmental, and additional sensor streams to operate successfully in multivariate settings [75]. As data volume and dimensionality have increased, new issues have evolved, requiring creative approaches and workable solutions. Examples include extracting deep features and spotting deep hidden patterns. Aside from being infrequently discussed in the literature, recurring anomalies make detection more difficult. A periodic subsequence abnormality is one that occurs on a regular basis [74]. The periodic subsequence anomaly detection technique can be used to spot recurring irregular transactions in fields like fraud detection.

D. ENERGY AND RESOURCE CONSTRAINTS

The resource limitations of IoT devices continue to be a significant barrier to the deployment of DL models – a process that requires sophisticated hardware. This is a significant difficulty because IoT devices and embedded systems typically have resource constraints. This means that using DL models may cause network failures and data disclosure while collecting and transmitting data to servers to be analyzed. When applying DL to real IoT systems, memory, and time efficiency would be two major challenges. Although DL models may be trained offline, there are still implementation issues. A persistent challenge is figuring out how to minimize the computing and storage resources required to run the DL model in resource-constrained applications. This may be because many DL models were not necessarily created for the IoT environment; investigations in this area will only continue until researchers get the optimal solution(s). In addition, [78] shows that designing highly accurate and resource-efficient deep learning models continues to be a challenge.

E. DEPLOYMENT ISSUES

Resource limitations have also exposed the relevance of DL models' adaptability, which is also a significant challenge. As devices and applications in the IoT ecosystem grow on a daily basis, so must DL's adaptability. In the real world, an IoT network is always open to attack from intruders and malicious systems. Following that, new devices are usually added to the IoT system. As additional devices join the network, the distribution of network traffic or signal frequency is likely to shift. Because a static-trained model cannot easily adjust to changing conditions, it may produce more false positives and false negatives. The request from the end user is another ever-changing factor. These modifications present new hurdles for DL applications in the IoT context. DL algorithms must deal with a rapidly changing environment from both a macro and micro perspective. Another factor to consider is that numerous IoT devices may be deployed in a variety of settings. The characteristics of the environment in which IoT is deployed may differ. Retraining a DL model for every setting takes a long time and also necessitates more labeled training data. In addition to the adaptability of models, IoT data analysis with DL may need to comply with industry-specific regulations (e.g., healthcare, finance) depending on the application. Ensuring that DL models meet these compliance requirements can be challenging. Deploying DL models on IoT devices or in the cloud requires careful planning and ongoing maintenance. Ensuring model updates, version control, and compatibility with evolving IoT environments is essential.

VII. CONCLUSION AND FUTURE WORKS

This study has provided a review showing different applications of DL for data prediction, anomaly detection, and correction in time series data generated from IoT. There are numerous use cases of the use of DL for IoT data analysis, in different fields (like agriculture, chemistry, healthcare, and so on). An in-depth look at the works reviewed shows that deep learning has worked in different areas with a high degree of success, but not without challenges. This review has highlighted some common challenges (such as model deployment issues, energy and resource constraints, privacy and security issues, and performance drawbacks in processing voluminous data with high dimensionality and anomaly interpretation) encountered when applying DL methodologies on IoT-generated time series data. As a result, a follow-up study to address these challenges is required. In addition, other facets of data processing and analysis should be studied. Nonetheless, this study contributes significantly to the existing body of knowledge regarding IoT data analysis, providing valuable insights for researchers.

REFERENCES

[1] R. Kumar, P. Kumar, and Y. Kumar, "Time series data prediction using IoT and machine learning technique," *Proc. Comput. Sci.*, vol. 167, pp. 373–381, Jan. 2020.

[2] M. Anasuri. (Apr. 23, 2021). *Time Series Data*. [Online]. Available: https://www.linkedin.com/pulse/time-series-data-platform-iot-mahesh-anasuri/

[3] A. Zhang, S. Song, J. Wang, and P. S. Yu, "Time series data cleaning: From anomaly detection to anomaly repairing," *Proc. VLDB Endowment*, vol. 10, no. 10, pp. 1046–1057, Jun. 2017.

[4] C. Wang, X. Huang, J. Qiao, T. Jiang, L. Rui, J. Zhang, R. Kang, J. Feinauer, K. A. McGrail, P. Wang, D. Luo, J. Yuan, J. Wang, and J. Sun, "Apache IoTDB: Time-series database for Internet of Things," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 2901–2904, Aug. 2020.

[5] T. J. Saleem and M. A. Chishti, "Deep learning for Internet of Things data analytics," *Proc. Comput. Sci.*, vol. 163, pp. 381–390, Jan. 2019.

[6] Snowflake. (2022). *What is IoT?* [Online]. Available: https://www.snowflake.com/guides/what-iot

[7] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[8] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-Things-based smart cities: Recent advances and challenges," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 16–24, Sep. 2017.

[9] S. Tian, W. Yang, J. M. L. Grange, P. Wang, W. Huang, and Z. Ye, "Smart healthcare: Making medical care more intelligent," *Global Health J.*, vol. 3, no. 3, pp. 62–65, Sep. 2019.

[10] B. Jovanovic. (May 13, 2022). *Internet of Things statistics for 2022—Taking Things Apart*. [Online]. Available: https://dataprot.net/statistics/iot-statistics/

[11] A. S. Rajawat, P. Bedi, S. B. Goyal, A. R. Alharbi, A. Aljaedi, S. S. Jamal, and P. K. Shukla, "Fog big data analysis for IoT sensor application using fusion deep learning," *Math. Problems Eng.*, vol. 2021, pp. 1–16, Oct. 2021.

[12] T. J. Saleem and M. A. Chishti, "Deep learning for the Internet of Things: Potential benefits and use-cases," *Digit. Commun. Netw.*, vol. 7, no. 4, pp. 526–542, Nov. 2021.

[13] A. Aldahiri, B. Alrashed, and W. Hussain, "Trends in using IoT with machine learning in health prediction system," *Forecasting*, vol. 3, no. 1, pp. 181–206, Mar. 2021.

[14] H. Hersbach et al., "The ERA5 global reanalysis," *Quart. J. Roy. Meteorol. Soc.*, vol. 146, no. 730, pp. 1999–2049, Jul. 2020.

[15] A. E. W. Johnson, T. J. Pollard, L. Shen, L.-W.-H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific Data*, vol. 3, no. 1, pp. 1–9, May 2016.

[16] J. Kierdorf, L. V. Junker-Frohn, M. Delaney, M. D. Olave, A. Burkart, H. Jaenicke, O. Müller, U. Rascher, and R. Roscher, "GrowliFlower: An image time-series dataset for growth analysis of cauliflower," *J. Field Robot.*, vol. 40, no. 2, pp. 173–192, Mar. 2023.

[17] (1987). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/datasets

[18] (Aug. 10, 2021). *Welcome to the Case Western Reserve University Bearing Data Center Website | Case School of Engineering | Case Western Reserve University*. [Online]. Available: https://engineering.case.edu/bearingdatacenter/welcome

[19] D. Puiu, P. Barnaghi, R. Tönjes, D. Kümper, M. I. Ali, A. Mileo, J. X. Parreira, M. Fischer, S. Kolozali, N. Farajidavar, F. Gao, T. Iggena, T.-L. Pham, C.-S. Nechifor, D. Puschmann, and J. Fernandes, "CityPulse: Large scale data analytics framework for smart cities," *IEEE Access*, vol. 4, pp. 1086–1108, 2016.

[20] (2022). *Pecan Street Dataport*. [Online]. Available: https://www.pecanstreet.org/dataport/

[21] (2019). *USGS Earthquake Hazards Program*. [Online]. Available: http://Earthquake.usgs.gov

[22] B. Veloso, R. P. Ribeiro, J. Gama, and P. M. Pereira, "The MetroPT dataset for predictive maintenance," *Sci. Data*, vol. 9, no. 1, Dec. 2022, Art. no. 764.

[23] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, 2014.

[24] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agricult.*, vol. 147, pp. 70–90, Apr. 2018.

[25] A. C. Mater and M. L. Coote, "Deep learning in chemistry," *J. Chem. Inf. Model.*, vol. 59, no. 6, pp. 2545–2559, 2019.

[26] A. Chahal and P. Gulia, "Deep learning: A predictive IoT data analytics method," *Int. J. Eng. Trends Technol.*, vol. 68, no. 7, pp. 25–33, Jul. 2020.

[27] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings Bioinf.*, vol. 18, no. 5, pp. 851–869, 2017.

[28] IBM Cloud Education. (May 1, 2020). *What is Deep Learning*. [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning #:~:text=Deep%20learning%20is%20a%20subset,from%20large%20 amounts%20of%20data

[29] S. Saxena. (Mar. 18, 2021). *Introduction to Long Short Term Memory (LSTM)*. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[30] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan. 2019.

[31] J. Schmidhuber, "The 2010s: Our decade of deep learning/outlook on the 2020s," AI Blog, IDSIA, Lugano, Switzerland, Tech. Rep., 2021.

[32] J. Schmidhuber, "The most cited neural networks all build on work done in my labs," AI Blog, IDSIA, Lugano, Switzerland, Tech. Rep., 2021.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, 2017, pp. 1–11.

[34] R. M. Farsani and E. Pazouki, "A transformer self-attention model for time series forecasting," *J. Electr. Comput. Eng. Innov.*, vol. 9, no. 1, pp. 1–10, 2021.

[35] J. Shi, M. Jain, and G. Narasimhan, "Time series forecasting (TSF) using various deep learning models," 2022, *arXiv:2204.11115*.

[36] J. Liang, "Getting started with machine learning," Tech. Rep., 2018.

[37] O. Eraliev and C.-H. Lee, "Performance analysis of time series deep learning models for climate prediction in indoor hydroponic greenhouses at different time intervals," *Plants*, vol. 12, no. 12, p. 2316, Jun. 2023.

[38] X. Song, Y. Guo, N. Li, and L. Zhang, "Integrated online prediction model for IoT data," *IEEE Signal Process. Lett.*, vol. 28, pp. 2043–2047, 2021.

[39] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, "Time-series representation learning via temporal and contextual contrasting," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1–8.

[40] Y. Inagaki, R. Shinkuma, T. Sato, and E. Oki, "Prioritization of mobile IoT data transmission based on data importance extracted from machine learning model," *IEEE Access*, vol. 7, pp. 93611–93620, 2019.

[41] A. Ed-Daoudy and K. Maalmi, "A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment," *J. Big Data*, vol. 6, no. 1, Dec. 2019, Art. no. 104.

[42] R. A. Kumar, J. V. Franklin, and N. Koppula, "A comprehensive survey on metaheuristic algorithm for feature selection techniques," *Mater. Today, Proc.*, vol. 64, pp. 435–441, Jan. 2022.

[43] K. Cao, T. Hu, Z. Li, G. Zhao, and X. Qian, "Deep multi-task learning model for time series prediction in wireless communication," *Phys. Commun.*, vol. 44, Feb. 2021, Art. no. 101251.

[44] S. Liu, L. Chen, X. Zhu, F. Yang, J. Li, and M. A. Diallo, "Prediction and early warning model of substation project cost based on data mining," in *Proc. Int. Conf. Multi-Modal Inf. Anal.*, 2022, pp. 400–407.

[45] R. Tkachenko, I. Izonin, N. Kryvinska, I. Dronyuk, and K. Zub, "An approach towards increasing prediction accuracy for the recovery of missing IoT data based on the GRNN-SGTM ensemble," *Sensors*, vol. 20, no. 9, p. 2625, May 2020.

[46] A. Akbar, A. Khan, F. Carrez, and K. Moessner, "Predictive analytics for complex IoT data streams," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1571–1582, Oct. 2017.

[47] S. Khaki, L. Wang, and S. V. Archontoulis, "A CNN-RNN framework for crop yield prediction," *Frontiers Plant Sci.*, vol. 10, Jan. 2020, Art. no. 1750.

[48] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Sci. Rep.*, vol. 8, no. 1, Apr. 2018, Art. no. 6085.

[49] S. Chae, S. Kwon, and D. Lee, "Predicting infectious disease using deep learning and big data," *Int. J. Environ. Res. Public Health*, vol. 15, no. 8, p. 1596, Jul. 2018.

[50] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, "ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis," *Future Gener. Comput. Syst.*, vol. 115, pp. 279–294, Feb. 2021.

[51] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of LSTM and BiLSTM in forecasting time series," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 3285–3292.

[52] M. Alhussein, K. Aurangzeb, and S. I. Haider, "Hybrid CNN-LSTM model for short-term individual household load forecasting," *IEEE Access*, vol. 8, pp. 180544–180557, 2020.

[53] H. Wei, H. Zhou, J. Sankaranarayanan, S. Sengupta, and H. Samet, "Residual convolutional LSTM for tweet count prediction," in *Proc. Companion Web Conf. Web Conf.*, 2018, pp. 1309–1316.

[54] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 114–119, Jun. 2019.

[55] D. P. Purbawa, R. Sarno, Malikhah, M. S. H. Ardani, S. I. Sabilla, K. R. Sungkono, C. Fatichah, D. Sunaryono, I. S. Parimba, and A. Bakhtiar, "Adaptive filter for detection outlier data on electronic nose signal," *Sens. Bio-Sens. Res.*, vol. 36, Jun. 2022, Art. no. 100492.

[56] K. A. P. D. Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. D. Albuquerque, "Internet of Things: A survey on machine learning-based intrusion detection approaches," *Comput. Netw.*, vol. 151, pp. 147–151, Mar. 2019.

[57] U. Sachdeva and P. R. Vamsi, "Analysis of deep learning models for anomaly detection in time series IoT sensor data," in *Proc. 14th Int. Conf. Contemp. Comput.*, Aug. 2022, pp. 54–62.

[58] Z. Ji, J. Gong, and J. Feng, "A novel deep learning approach for anomaly detection of time series data," *Sci. Program.*, vol. 2021, pp. 1–11, Jul. 2021.

[59] X. Xie, D. Wu, S. Liu, and R. Li, "IoT data analytics using deep learning," 2017, *arXiv:1708.03854*.

[60] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021.

[61] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Anchorage, AK, USA, Jul. 2019, pp. 2828–2837.

[62] A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using recurrent neural networks (RNN)," in *Proc. Integr. Commun. Navigat. Surveill. (ICNS)*, Herndon, VA, USA, Apr. 2016, pp. 5C2-1–5C2-8.

[63] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts, "Anomaly detection for time series using VAE-LSTM hybrid model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Barcelona, Spain, May 2020, pp. 4322–4326.

[64] Y. Wang, X. Du, Z. Lu, Q. Duan, and J. Wu, "Improved LSTM-based time-series anomaly detection in rail transit operation environments," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 9027–9036, Dec. 2022.

[65] F. Zeng, M. Chen, C. Qian, Y. Wang, Y. Zhou, and W. Tang, "Multivariate time series anomaly detection with adversarial transformer architecture in the Internet of Things," *Future Gener. Comput. Syst.*, vol. 144, pp. 244–255, Jul. 2023.

[66] J. Kim, H. Kang, and P. Kang, "Time-series anomaly detection with stacked transformer representations and 1D convolutional network," *Eng. Appl. Artif. Intell.*, vol. 120, Apr. 2023, Art. no. 105964.

[67] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *Proc. ICLR*, 2022, pp. 1–20.

[68] W. Li, C. Feng, T. Chen, and J. Zhu, "Robust learning of deep time series anomaly detection models with contaminated training data," 2022, *arXiv:2208.01841*.

[69] Y. Li, X. Peng, J. Zhang, Z. Li, and M. Wen, "DCT-GAN: Dilated convolutional transformer-based GAN for time series anomaly detection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3632–3644, Apr. 2023.

[70] M. Ma, L. Han, and C. Zhou, "BTAD: A binary transformer deep neural network model for anomaly detection in multivariate time series data," *Adv. Eng. Informat.*, vol. 56, Apr. 2023, Art. no. 101949.

[71] L. Xu, K. Xu, Y. Qin, Y. Li, X. Huang, Z. Lin, N. Ye, and X. Ji, "TGAN-AD: Transformer-based GAN for anomaly detection of time series data," *Appl. Sci.*, vol. 12, no. 16, p. 8085, Aug. 2022.

[72] T. Lin, "Deep learning for IoT," Tech. Rep., 2021.

[73] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings Bioinf.*, 2016.

[74] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," 2022, *arXiv:2211.05244*.

[75] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for IoT time-series data: A survey," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6481–6494, Jul. 2020.

[76] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, "A review of deep learning models for time series prediction," *IEEE Sensors J.*, vol. 21, no. 6, pp. 7833–7848, Mar. 2021.
[77] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120043–120065, 2021.
[78] K. Lakshmanna, R. Kaluri, N. Gundluru, Z. S. Alzamil, D. S. Rajput, A. A. Khan, M. A. Haq, and A. Alhussen, "A review on deep learning techniques for IoT data," *Electronics*, vol. 11, no. 10, p. 1604, May 2022.

POLYCARP SHIZAWALIYI YAKOI received the B.Sc. and M.Sc. degrees from Cyprus International University, in 2016 and 2018, respectively. From 2016 to 2018, he was a Research Assistant with the Faculty of Engineering, Cyprus International University. His current research interests include deep learning, time series, particle swarm optimization, and the Internet of Things.

XIANGFU MENG was born in 1981. He received the Ph.D. degree from Northeastern University, in 2010. He is currently a Full Professor and a Ph.D. Supervisor with Liaoning Technical University. His current research interests include spatial data management, recommender systems, and artificial intelligence.

SHUOLIN CUI was born in 1998. He received the bachelor's degree from Xidian University, China. He is currently an Academic Visiting Scholar with Liaoning Technical University. His current research interests include machine learning and digital image processing.

DANLADI SULEMAN received the M.Sc. degree in management information systems from Cyprus International University, in 2020. He is currently pursuing the Ph.D. degree with the University of the Sunshine Coast, Australia. His current research interests include the IoT network security, sentiment analysis, smart transportation, and digital government.

XUEYONG YANG is currently pursuing the master's degree with the Liaoning University of Engineering and Technology (Liaoning Technical University). His current research interest includes tumor mutation burden value prediction based on histopathological images.