Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things

Yi Liu
City University of Hong Kong
47 PUBLICATIONS **1,599** CITATIONS

Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things

Yi Liu[1], Neeraj Kumar[2], Zehui Xiong[3], Wei Yang Bryan Lim[3], Jiawen Kang[3,†], Dusit Niyato[3], *Fellow, IEEE*
[1]*School of Data Science of Technology, Heilongjiang University, Harbin, China*
[2]*Department of Electrical and Instrumentation Engineering, Thapar Institute of Engineering & Technology, Patiala, India*
[3]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*
{97liuyi, neeraj.kumar.in}@ieee.org, {ZXIONG002, limw0201}@e.ntu.edu.sg, {†kavinkang, dniyato}@ntu.edu.sg

***Abstract*—With the rapid development of the Industrial Internet of Things (IIoT), various IoT devices and sensors generate massive industrial sensing data. Sensing big data can be analyzed for insights that lead to better decisions and strategic industrial production by using advanced machine learning technologies. However, vulnerable IoT devices are easy to be compromised thus causing IoT devices failures (i.e., anomalies). The anomalies seriously affect the production of industrial products, thereby, it is increasingly important to accurately and timely detect anomalies. To this end, we first introduce a Federated Learning (FL) framework to enable decentralized edge devices to collaboratively train a Deep Anomaly Detection (DAD) model, which can improve its generalization ability. Second, we propose a Convolutional Neural Network-Long Short Term Memory (CNN-LSTM) model to accurately detect anomalies. The CNN-LSTM model uses CNN units to capture fine-grained features and retains the advantages of LSTM unit in predicting time series data. Third, to achieve real-time and lightweight anomaly detection in the proposed framework, a gradient compression mechanism is applied to reduce communication costs and improve communication efficiency. Extensive experiment results based on real-world datasets demonstrate that the proposed framework and mechanism can accurately and timely detect anomalies, and also reduce about 50% communication overhead when compared with traditional schemes.**

***Index Terms*—Federated learning, anomaly detection, gradient compression, industrial internet of things**

I. Introduction

With widespread deployment of powerful sensors and Internet of Things (IoT) devices in the Industrial IoT (IIoT), massive sensing data is becoming industrial big data and brings the high potential for industrial production and manufacturing processes. By integrating with industrial big data with resource-rich edge computing, a lot of promising applications have emerged, such as smart manufacturing, intelligent transportation, and intelligent logistics [1]. The edge devices provide powerful computation resources to enable real-time, flexible, and quick decision making based on the industrial big data analysis for the IIoT applications, which has greatly promoted the development of Industry 4.0 [2]. However, the IIoT applications are suffering from critical security risks caused by abnormal IIoT nodes which hinders the rapid development of IIoT. For example, in smart manufacturing scenarios, industrial devices acting as IIoT nodes, e.g., engines with sensors, that have abnormal behaviors (e.g., abnormal traffic and irregular reporting frequency) may cause industrial production interruption thus resulting in huge economic losses for factories [3]. Edge devices (e.g., industrial robots), generally collect sensing data from IIoT nodes, especially time-series data, to analyze and capture the behaviors and operating condition of IIoT nodes by edge computing [4]. Therefore, these sensing time series data can be used to detect the anomaly behaviors of IIoT nodes [5].

To solve the abnormality problems from IIoT devices, typical methods are to perform abnormal detection for the affected IIoT devices [6]. Previous work focused on utilizing deep anomaly detection (DAD) [7] approaches to detect abnormal behaviors of IIoT devices by analyzing sensing time series data. DAD techniques can learn hierarchical discriminative features from historical time-series data. In [8], the authors proposed a Long Short Term Memory (LSTM) networks-based deep learning model to achieve anomaly detection in sensing time series data. Munir *et al.* in [9] proposed a novel DAD approach, called DeepAnT, to achieve anomaly detection by utilizing deep Convolutional Neural Network (CNN) to predict anomaly value. Although the existing DAD approaches have achieved success in anomaly detection, they cannot be directly applied to the IIoT scenarios with distributed edge devices for timely and accurate anomaly detection. The reasons are two-fold: (i) the most of detection models are not flexible enough in traditional approaches, the edge devices lack dynamic and automatically updated detection models for different scenarios, and hence the models fail to accurately predict frequently updated time-series data [6]; (2) due to privacy concerns, the edge devices are not willing to share their own collected time-series data with each other, thus the data exists in the form of "islands." The data islands significantly degrade the performance of anomaly detection.

To address the above challenges, a promising on-device privacy-preserving distributed machine learning paradigm, called federated learning (FL) [10], has been proposed for edge devices to train a global DAD model while keeps the training datasets locally without sharing raw training data. Such a framework allows edge devices to collaboratively train an on-device DAD model without compromising privacy. For example, reference [11] utilized FL framework to propose a Back-propagation Neural Networks (i.e., BPNNs) based approaches for anomaly detection. However, previous researches ignore the communication overhead during model training by using

FL among large-scale edge devices. Expensive communication overhead can cause excessive overhead and long convergence time for edge devices, and consequently the on-device DAD model cannot promptly detect anomalies. Therefore, it is necessary to develop a communication-efficient FL framework to achieve accurate and timely anomaly detection for edge devices.

In this paper, we propose a communication-efficient FL framework that leverages a CNN-LSTM model to achieve accurate and timely anomaly detection for edge devices. First, we introduce an FL framework to enable distributed edge devices to collaboratively train a global DAD model without compromising privacy. Second, we propose a CNN-LSTM model to detect anomalies. Specifically, we use CNNs to extract fine-grained features of historical observation-sensing time-series data and use LSTM modules for time-series prediction. Such a model can prevent memory loss and gradient dispersion problems [7]. Third, to further improve the communication efficiency of the proposed framework, we propose a gradient compression scheme based on Top-*k* selection to reduce the number of gradients uploaded by edge devices. We evaluate the proposed framework on two real-world datasets: power demand and the space shuttle. Experimental results show that the proposed framework can achieve high-efficiency communication and achieve accurate and timely anomaly detection. The contributions of this paper are summarized as follows:

- We introduce an FL framework to develop an on-device collaborative deep anomaly detection model for edge devices in IIoT.
- We propose a CNN-LSTM model to detect anomalies, which uses CNN to capture the fine-grained features of time series data and uses LSTM module to accurately and timely detect anomalies.
- We propose a Top-*k* selection-based gradient compression scheme to improve the proposed framework's communication efficiency. Such a scheme decreases communication overhead by reducing the exchanged gradient parameters between the edge devices and the cloud aggregator.
- We conduct extensive experiments on two real-world datasets to demonstrate that the proposed framework can accurately detect anomalies with low communication overhead.

II. Preliminary

A. Anomalies in Industrial Big Data

In statistics, anomalies (also called outliers and abnormalities) are the data points that are significantly different from other observations of big data [7]. We assume that N_1, N_2, and N_3 are regions composed of most observations, so they are regarded as normal data instance regions. If data points O_1 and O_2 are far from these regions, they can be classified as anomalies. To define anomalies more formally, we assume that an *n*-dimensional dataset $\vec{x}_i = (x_{i,1}, \ldots, x_{i,n})$ follows a normal distribution and its mean μ_{j} and variance σ_j for each dimension where $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. Specifically, for $j \in \{1, \ldots, n\}$, under the assumption of the normal distribution, we have $\mu_j = \sum_{i=1}^{m} x_{i,j}/m$, $\sigma_j^2 = \sum_{i=1}^{m} (x_{i,j} - \mu_j)^2/m$, if there is a new vector $\vec{x}$, the probability $p(\vec{x})$ of anomaly can be calculated as follows:

$$p(\vec{x}) = \prod_{j=1}^{n} p\left(x_j; \mu_j, \sigma_j^2\right) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right). \tag{1}$$

We can then judge whether vector $\vec{x}$ belongs to an anomaly according to the probability value for industrial big data [7].

B. Federated Learning

Federated Learning (FL) is a promising distributed machine learning framework for privacy protection proposed by Google [10]. The procedure of FL is divided into three phases: the initialization, the aggregation, and the update phase. In the initialization phase, we consider that FL with N edge devices and a cloud aggregator, distributes a pre-trained global model ω_t on the public datasets (e.g., MNIST dataset) to each edge devices. Following that, each device uses local dataset $\mathcal{D}_k$ of size D_k to train and improve the current global model ω_t in each iteration. In the aggregation phase, the cloud aggregator collects local gradients uploaded by the edge nodes (i.e., edge devices). To do so, the local loss function to be optimized is defined as follows: $\min\limits_{x \in \mathbb{R}^d} F_k(x) = \frac{1}{D_k} \sum_{i \in D_k} \mathbb{E}_{z_i \sim D_k} f(x; z_i) + \lambda h(x)$, where $f(\cdot;\cdot)$ is the local loss function for edge device k, $\forall \lambda \in [0, 1]$, $h(\cdot)$ is a regularizer function for edge device k, and $\forall i \in [1, \ldots, n], z_i$ is sampled from the local dataset $\mathcal{D}_k$ on the k device. In the update phase, the cloud aggregator uses Federated Averaging (FedAVG) algorithm [10] to obtain a new global model ω_{t+1} for the next iteration. Therefore, we have $\omega_{t+1} \leftarrow \omega_t + \frac{1}{n} \sum\limits_{n=1}^{N} F_{t+1}^{n}$, where $\frac{1}{n} \sum\limits_{n=1}^{N} F_{t+1}^{n}$ denotes the average aggregation. Both the edge devices and the cloud aggregator repeat the above process until the global model reaches convergence.

C. Gradient Compression for Federated Learning

Large-scale FL training requires significant communication bandwidth for gradient exchange, which limits the scalability of multi-nodes training [12]. In this context, Lin *et al.* in [12] stated that 99.9% of the gradient exchange in D-SGD is redundant. To avoid expensive communication bandwidth limiting large-scale distributed training, gradient compression is proposed to greatly reduce communication bandwidth. Researchers generally use gradient quantization [13] and gradient sparsification [12] to achieve gradient compression. Gradient quantization reduces communication bandwidth by quantizing gradients to low-precision values. Gradient sparsification uses threshold quantization (i.e., sends only gradients larger than a predefined constant threshold) to reduce communication bandwidth.

III. System Model

In this paper, we propose an on-device communication-efficient FL-based deep anomaly detection framework that

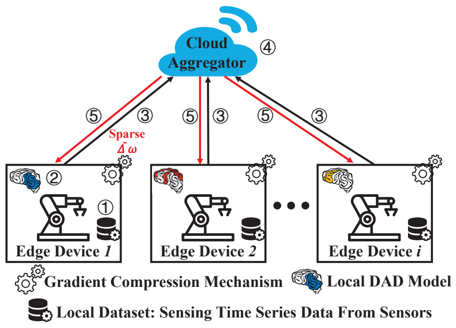

Fig. 1. The overview of the communication-efficient FL-based DAD framework in IIoT: (i) The edge device uses the sensing time series data collected from IIoT nodes as a local dataset (i.e., sensing time series data from IIoT nodes) (step ①). (ii) The edge device performs local model training on the local dataset (step ②). (iii) The edge device uploads the sparse gradients $\tilde{\Delta}\omega$ to the cloud aggregator by using a gradient compression mechanism (step ③). (iv) The cloud aggregator obtains a new global model by aggregating the sparse gradients uploaded by the edge device (step ④). (v) The cloud aggregator sends the new global model to each edge device. (step ⑤).

involves multiple edge devices for collaborative model training in IIoT, as illustrated in Fig. 1. In particular, this framework consists of a cloud aggregator and edge devices. Furthermore, the proposed framework also includes two mechanisms: an anomaly detection mechanism and a gradient compression mechanism. More details are described as follows:

- ***Cloud Aggregator***: The cloud aggregator is generally a cloud server with strong computing power and rich computing resources. The cloud aggregator contains two functions: (1) initializes the global model and sends the global model to the all edge devices; (2) aggregates the gradients uploaded by the edge devices until the model converges.
- ***Edge Devices***: Edge devices are generally agents and clients, such as whirlpool, wind turbine, and vehicle, which contain local models and functional mechanisms. Each edge device uses the local dataset to train the global model sent by the cloud aggregator and uploads the gradients to the cloud aggregator until the global model converges.
- ***Deep Anomaly Detection Mechanism:*** The deep anomaly detection mechanism is deployed in the edge devices, which can detect anomalies to reduce economic losses.
- ***Gradient Compression Mechanism:*** The gradient compression mechanism is deployed in the edge devices, which can compress the local gradients to reduce the number of gradients exchanged between the edge devices and the cloud aggregator, thereby reducing communication overhead.

IV. A Communication-Efficient Deep Anomaly Detection Framework

A. *CNN-LSTM Model*

We present a CNN-LSTM model including an input layer, a CNN unit, an LSTM unit, and an output layer, as shown in Fig. 2. First, we use the preprocessed data as input to the input layer. Second, we use CNN to capture the fine-grained features of the inputs. Third, we use the output of the CNN unit as the input of LSTM unit, and utilize LSTM to predict future time series data. Finally, we propose an anomaly detection score to detect anomalies.

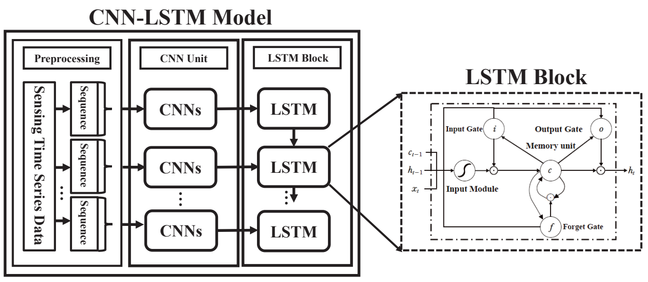

Fig. 2. The overview of the attention mechanism-based CNN-LSTM Model.

Preprocessing: We normalize the sensing time series data collected by the IIoT nodes into [0,1] to accelerate the model convergence.

CNN Unit: We use CNN unit to extract fine-grained features of time series data. The CNN module is formed by stacking multiple layers of one-dimensional (1-D) CNN, and each layer includes a convolution layer, a batch normalization layer, and a non-linear layer. Such modules implement sampling aggregation by using pooling layers and create hierarchical structures that gradually extract more abstract features through the stacking of convolutional layers. This module outputs m feature sequences of length n, and the size can be expressed as $(n \times m)$. The feature aggregation part uses the stacking of multiple convolutions and pooling layers to extract key features from the sequence and uses a convolution kernel of size 1×1 to mine the linear relationship. The scale restoration part restores the key features to $(n \times m)$, which is consistent with the size of the output features of CNN module, and then uses the sigmoid function to constrain the values to [0,1].

LSTM Block: We next use a variant of a recurrent neural network, called LSTM, to support accurately prediction the sensing time series data to detect anomalies, as shown in Fig. 2. LSTM uses a well-designed “gate” structure to remove or add information to the state of the cell. The “gate” structure is a method of selectively passing information. LSTM cells include forget gates f_t, input gates i_t, and output gates o_t. The calculations on the three gate structures are defined as follows:

$$\begin{aligned}
f_t &= \sigma_l(W_f \cdot [h_{t-1}, x_t] + b_f), \\
i_t &= \sigma_l(W_i \cdot [h_{t-1}, x_t] + b_i), \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t, \\
o_t &= \sigma_l(W_o \cdot [h_{t-1}, x_t] + b_o), \\
h_t &= o_t * \tanh(C_t),
\end{aligned} \tag{2}$$

where W_f, W_i, W_C, W_o, and b_f, b_i, b_C, b_o are the weight matrices and the bias vectors for input vector x_t at time step t, respectively. σ_l is the activation function, $*$ represents element-wise multiplication of a matrix, C_t represents the cell state,

Algorithm 1: Gradient compression mechanism on edge node k.

```
Input: $\mathcal{G} = \{g^1, g^2, \ldots, g^k\}$ is the edge node's gradient, $B$ is the local mini-batch size, $\mathcal{D}_k$ is the local dataset, $\eta$ is the learning rate, $f(\cdot,\cdot)$ is the edge node's loss function, and the optimization function SGD.
Output: Parameter $\omega$.
1  Initialize parameter $\omega_t$;
2  $g^k \leftarrow 0$;
3  for $t = 0, 1, \cdots$ do
4      $g_t^k \leftarrow g_{t-1}^k$;
5      for $i = 1, 2, \cdots$ do
6          Sample data $x$ from $\mathcal{D}_k$;
7          $g_t^k \leftarrow g_{t-1}^k + \frac{1}{|\mathcal{D}_k|B}\nabla f(x;\omega_t)$;
8  if Gradient Clipping then
9      $g_t^k \leftarrow$ Local_Gradient_Clipping $(g_t^k)$;
10 foreach $g_t^{k_j} \in \{g_t^k\}$ and $j = 1, 2, \cdots$ do
11     $\mathrm{Thr} \leftarrow |\mathrm{Top}\,\rho\%\ of\ \{g_t^k\}|$;
12     if $|g_t^{k_j}| \geq \mathrm{Thr}$ then
13         Send this gradient to the cloud aggregator;
14     if $|g_t^{k_j}| < \mathrm{Thr}$ then
15         The edge node $k$ uses the local gradient accumulation scheme to accumulate gradients until the gradient reaches Thr;
16     Aggregate $g_t^k : g_t \leftarrow \sum_{k=1}^{N}(\mathrm{sparse}\,\tilde{g}_t^k)$;
17     $\omega_{t+1} \leftarrow \mathrm{SGD}(\omega_t, g_t)$.
18 return $\omega$.
```

h_{t-1} is the state of the hidden layer at time step $t-1$, and h_t is the state of the hidden layer at time step t.

Anomaly Detection: We use CNN-LSTM model to predict real-time and future sensing time series data in different edge devices:

$$[x^i_{n-T+1}, x^i_{n-T+2}, \ldots, x^i_n] \xrightarrow{f(\cdot)} [x^i_{n+1}, x^i_{n+2}, \ldots, x^i_{n+T}], \quad (3)$$

where $f(\cdot)$ is the prediction function. We use LSTM unit for time series prediction. We use anomaly scores for anomaly detection, which is defined as follows:

$$A_n = (\beta_n - \mu)^T \sigma^{-1} (\beta_n - \mu), \quad (4)$$

where A_n is the anomaly score, $\beta_n = |x^i_n - x'^i_n|$ is the reconstruction error vector, and the error vectors β_n for the time series in the sequences X^i are used to estimate the parameters μ and σ of a Normal distribution $\mathcal{N}(\mu;\sigma)$ using Maximum Likelihood Estimation. In an unsupervised setting, when $A_n \geq \varsigma$ ($\varsigma = \max F_\theta = \frac{(1+\theta^2)\times P \times R}{\theta^2 P + R}$), where P is precision, R is recall, and θ is the parameter, a point in a sequence can be predicted to be "anomalous", and otherwise "normal".

B. Gradient Compression Mechanism

In the context of deep learning, the sparseness of the gradient during model training can generally reach 99.9%, only the 0.1% gradients with the largest absolute value are useful for gradients aggregation [14]. Inspired by the above facts, we propose a gradient compression mechanism to reduce the number of gradients exchanged between the cloud aggregator and the edge devices. Since the edge devices only send some gradients with large absolute values to the cloud aggregator, the edge devices use the local gradient accumulation scheme to accumulate the small gradients in the edge device.

When the gradients' sparsification reaches a high value (e.g., 99%), it will affect the model convergence. By following [14], we use momentum correction and local gradient clipping to mitigate this effect. Momentum correction can make the accumulated small local gradients converge toward the gradients with a larger absolute value, thereby accelerating the model's convergence speed. Local gradient clipping is used to alleviate the problem of gradient explosions [14]. Next, we prove that local gradient accumulation scheme will not affect the model convergence: We assume that $g^{(i)}$ is the i-th gradient, $u^{(i)}$ denotes the sum of the gradients using the aggregation algorithm in [10], $v^{(i)}$ denotes the sum of the gradients using the local gradient accumulation scheme, and m is the rate of gradient descent. If the i-th gradient does not exceed threshold until the $(t-1)$-th iteration and triggers the model update, we have:

$$u^{(i)}_{t-1} = m^{t-2} g^{(i)}_1 + \cdots + m g^{(i)}_{t-2} + g^{(i)}_{t-1}, \quad (5)$$

$$v^{(i)}_{t-1} = \left(1 + \cdots + m^{t-2}\right) g^{(i)}_1 + \cdots + (1+m) g^{(i)}_{t-2} + g^{(i)}_{t-1}, \quad (6)$$

then we can update $\omega^{(i)}_t = w^{(i)}_1 - \eta \times v^{(i)}_{t-1}$ and set $v^{(i)}_{t-1} = 0$. If the i-th gradient reaches the threshold at the t-th iteration, model update is triggered, thus we have:

$$u^{(i)}_t = m^{t-1} g^{(i)}_1 + \cdots + m g^{(i)}_{t-1} + g^{(i)}_t, \quad (7)$$

$$v^{(i)}_t = m^{t-1} g^{(i)}_1 + \cdots + m g^{(i)}_{t-1} + g^{(i)}_t. \quad (8)$$

Then we can update $\omega^{(i)}_{t+1} = \omega^{(i)}_t - \eta \times v^{(i)}_t = \omega_1{}^{(i)} - \eta \times \left[\left(1 + \cdots + m^{t-1}\right) g^{(i)}_1 + \cdots + (1+m) g^{(i)}_{t-1} + g^{(i)}_t\right] = w^{(i)}_1 - \eta \times v^{(i)}_{t-1}$, so the result of using the local gradient accumulation scheme is consistent with the effect of using the aggregation algorithm in [10].

The specific implementation phases of the gradient compression mechanism are given as follows:

i) ***Phase 1, Local Training:*** Edge devices use the local dataset to train the local model. Each device uploads the local updated gradients to the cloud aggregator. In particular, we use the gradient accumulation scheme to accumulate local small gradients.

ii) ***Phase 2, Gradient Compression:*** Each edge device uses Algorithm 1 to compress the gradients and upload sparse gradients (i.e., **only gradients larger than a threshold are transmitted.**) to the cloud aggregator. Note that if the gradients are smaller than threshold, the edge devices use the local gradient accumulation to accumulate the gradients until the gradients' value reaches the threshold.

iii) ***Phase 3, Gradient Aggregation:*** The cloud aggregator obtains the global model by aggregating sparse gradients and sends this global model to the edge devices.

The gradient compression algorithm is thus presented in Algorithm 1.

V. Experiments

In this section, the proposed framework is applied to two real-world datasets, i.e., power demand[1] and space shuttle[2] (see the link for more details). These datasets are time series data collected by IIoT nodes in edge devices. We divide all datasets into a training set and a test set in a 7: 3 ratio. We implement the proposed framework by using Pytorch and PySyft. The experiment is conducted on a virtual workstation with the Ubuntu 18.04 operation system, Intel (R) Core (TM) i5-4210M CPU, 16GB RAM, 512GB SSD.

A. Evaluation Setup

In this experiment, to determine the hyperparameter ρ of the gradient compression mechanism, we first apply a simple CNN network (i.e., CNN with 2 convolutional layers followed by 1 fully connected layer) in the proposed framework to perform the classification task on MNIST and CIFAR-10 dataset. The pixels in all datasets are normalized into [0,1]. During the simulation, the number of edge devices is $N = 10$, the learning rate is $\eta = 0.001$, the training epoch is $E = 1000$, the mini-batch size is $B = 128$, and we follow reference [15] and set θ as 0.05. We adopt Root Mean Square Error (RMSE) to indicate the performance of CNN-LSTM model as follows:

$$\text{RMSE} = [\frac{1}{n}\sum_{i=1}^{n}(|y_i - \hat{y}_p|)^2]^{\frac{1}{2}}, \tag{9}$$

where y_i is the observed sensing time series data, and $\hat{y}_p$ is the predicted sensing time series data.

B. Structure of the Proposed Framework

In the context of deep gradient compression scheme, proper hyperparameter selection, i.e., a threshold of absolute gradient value, is a notable factor that determines the proposed framework performance. In this section, we investigate the performance of the proposed framework with different thresholds and try to find a best-performing threshold for it. In particular, we employ $\rho \in \{0.1, 0.2, 0.3, 0.5, 0.9, 1, 100\}$ to adjust the best threshold of the proposed framework. We use MNIST and CIFAR-10 datasets to evaluate the performance of the proposed framework with the selected threshold. As shown in Fig. 3, we observe that the larger ρ, the better the performance of the proposed framework. For MNIST task, the results show that when $\rho = 0.3$, the accuracy is 97.25%; when $\rho = 100$, the accuracy is 99.08%. This means that the model increases gradient size by about 300 times, but the accuracy is only improved by 1.83%. However, the higher the selected threshold, the more the gradient of the exchange, which means that the communication overhead is also greater. Therefore, to achieve a good trade-off between the gradient threshold and communication overhead, we choose $\rho = 0.3$ as the best threshold of our scheme.

[1]https://archive.ics.uci.edu/ml/datasets/

[2]https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)

C. Performance of the Proposed Framework

We compare the performance of the proposed model with that of LSTM [15], Gate Recurrent Unit (GRU) [16], Stacked Auto Encoders (SAEs) [17], and Support Machine Vector (SVM) [18] method with an identical simulation configuration. All models are developed based on FL framework and are popular DAD models for general anomaly detection applications. We evaluate these models on two real-world datasets, i.e., power demand and space shuttle. In these datasets, there are normal time series and abnormal time series data, where the abnormal time series indicate that the edge device has a certain failure. Therefore, we can detect abnormal time series data by using the proposed model to calculate the abnormal score.

First, we compare the accuracy of the proposed model with the baseline methods in anomaly detection. We determine the $\max F_\theta$ and hyperparameter ς based on the accuracy and recall of the model on the training set. The hyperparameters ς of the dataset power demand and space shuttle are 0.75 and 0.80. In Fig. 4(a), experimental results show that the proposed model achieves the highest accuracy on all two datasets. For example, for the dataset power demand, the accuracy of CNN-LSTM model is 94.23%, which is 5.25% higher than that of SVM model. From the experimental results, CNN-LSTM has better robustness to different datasets. The reason is that we use the FL framework to train and update the model, which can learn the time-series features from different edge devices as much as possible, thereby improving the robustness of the model. Furthermore, the FL framework provides opportunities for edge devices to update models in a timely manner. This helps the edge device owner to update the model on the edge devices in time.

Second, we need to evaluate the prediction error of the proposed model and the baseline methods. As shown in Fig. 4(b), experimental results show that the proposed model achieves the best performance on two real-world datasets. For the Space Shuttle dataset, RMSE of CNN-LSTM model is 57.7% lower than that of SVM model. The reason is that CNN-LSTM model uses CNN units to capture fine-grained features and prevent memory loss and gradient dispersion problems. Memory loss and gradient dispersion problems often occur in encoder-decoder models such as LSTM and GRU models. Furthermore, the proposed model retains the advantages of LSTM unit in predicting time series data. Therefore, the proposed model can not only accurately detect abnormalities, but also accurately predict time series data.

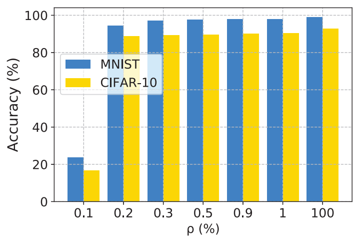

Fig. 3. The accuracy of the proposed framework with different ρ on MNIST and CIFAR-10 datasets.

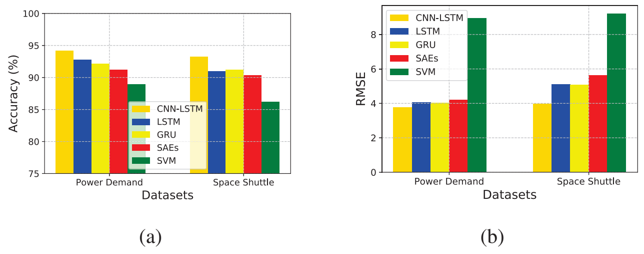

Fig. 4. Performance comparsion of detection accuracy for CNN-LSTM, CNN-LSTM, LSTM, GRU, SAEs, and SVM on different datasets. (a) Accuracy; (b) RMSE.

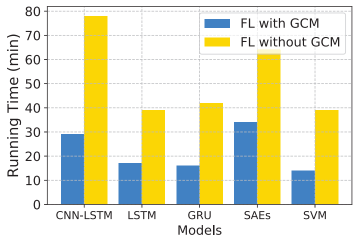

Fig. 5. Comparison of communication efficiency between FL with GCM and FL without GCM with different models.

D. Communication Efficiency of the Proposed Framework

We then compare the communication efficiency between the FL framework with the gradient compression mechanism (GCM) and FL framework without GCM. We apply the same model (i.e., CNN-LSTM, LSTM, GRU, SAEs, and SVM) in the framework. Note that we fix the communication overhead of each round, so we can use the running time of the model to compare the communication efficiency. As shown in Fig. 5, we observe that the running time of FL framework with GCM is about 50% that of the framework without GCM. The reason is that GCM can reduce the number of gradients exchanged between the edge devices and the cloud aggregator. In Section IV-B, we show that GCM can compress the gradient by 300 times without compromising the accuracy. Therefore, the proposed communication efficient framework is practical and effective in real-world applications.

VI. Conclusion

In this paper, to ensure industrial application security, effective anomaly detection mechanisms based on sensing big data are becoming more and more important. Therefore, a novel communication-efficient FL-based deep anomaly detection framework is proposed for industrial edge devices. We then propose an FL-based CNN-LSTM model to detect anomalies in IIOT. To further improve the communication efficiency of the proposed model, a gradient compression mechanism based on Top-k selection is applied to reduce communication costs. Finally, we evaluate the performance of the proposed model and mechanisms on real-world datasets and also compare performance with existing methods. Numerical results show that the proposed CNN-LSTM model can achieve the highest accuracy on all datasets. The proposed mechanism can significantly improve communication efficiency through compressing gradients by 300 times without losing accuracy.

References

[1] H. Peng and X. Shen, “Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks,” *IEEE Transaction on Network Science and Engineering*, to appear.

[2] Y. Wu, Y. Liu, S. H. Ahmed, J. Peng, and A. A. Abd El-Latif, “Dominant data set selection algorithms for electricity consumption time-series data analysis based on affine transformation,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4347–4360, 2020.

[3] Y. Peng, A. Tan, J. Wu, and Y. Bi, “Hierarchical edge computing: A novel multi-source multi-dimensional data anomaly detection scheme for industrial internet of things,” *IEEE Access*, vol. 7, pp. 111 257–111 270, 2019.

[4] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “Lstm-based encoder-decoder for multi-sensor anomaly detection,” *arXiv preprint arXiv:1607.00148*, 2016.

[5] T. Luo and S. G. Nagarajan, “Distributed anomaly detection using autoencoder neural networks in wsn for iot,” in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[6] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “Dïot: A federated self-learning anomaly detection system for iot,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[7] R. Chalapathy and S. Chawla, “Deep learning for anomaly detection: A survey,” *arXiv preprint arXiv:1901.03407*, 2019.

[8] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015.

[9] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, “Deepant: A deep learning approach for unsupervised anomaly detection in time series,” *IEEE Access*, vol. 7, pp. 1991–2005, 2018.

[10] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.

[11] M. Tsukada, M. Kondo, and H. Matsutani, “A neural network based on-device learning anomaly detector for edge devices,” *arXiv preprint arXiv:1907.10147*, 2019.

[12] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, “Deep gradient compression: Reducing the communication bandwidth for distributed training,” *arXiv preprint arXiv:1712.01887*, 2017.

[13] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, “Adacomp: Adaptive residual gradient compression for data-parallel distributed training,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] J. Wangni, J. Wang, J. Liu, and T. Zhang, “Gradient sparsification for communication-efficient distributed optimization,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1299–1309.

[15] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “Lstm-based encoder-decoder for multi-sensor anomaly detection,” *arXiv preprint arXiv:1607.00148*, 2016.

[16] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li, “Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach,” in *Asian Conference on Machine Learning*, 2018, pp. 97–112.

[17] N. Chouhan, A. Khan *et al.*, “Network anomaly detection using channel boosted and residual learning based deep convolutional neural network,” *Applied Soft Computing*, vol. 83, p. 105612, 2019.

[18] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, “High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning,” *Pattern Recognition*, vol. 58, pp. 121–134, 2016.