Detecting Cyberattacks using Anomaly Detection in Industrial Control Systems: A Federated Learning approach

Thu Huong Truong, Ta Phuong Bac, Dao Minh Long, Tran Duc Luong, Nguyen Minh Dan, Le Anh Quang, Le Thanh Cong, Bui Doan Thang, Kim Phuc Tran

Detecting Cyberattacks using Anomaly Detection in Industrial Control Systems: A Federated Learning approach

Truong Thu Huong*

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam.

Ta Phuong Bac

School of Electronic Engineering, Soongsil University, Seoul, Korea.

Dao Minh Long

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam.

Tran Duc Luong

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam.

Nguyen Minh Dan

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam.

Le Anh Quang

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam.

Le Thanh Cong

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam.

Bui Doan Thang

School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam.

Tran Kim Phuc

Université de Lille, ENSAIT, GEMTEX, F-59000 Lille, France

*Corresponding Author: Truong Thu Huong
Email address: huong.truongthu@hust.edu.vn (Truong Thu Huong)

[1]Since 1880.

Abstract

In recent years, the rapid development and wide application of advanced technologies have profoundly impacted industrial manufacturing, leading to smart manufacturing (SM). However, the Industrial IoT (IIoT)-based manufacturing systems are now one of the top industries targeted by a variety of attacks. In this research, we propose detecting Cyberattacks in Industrial Control Systems using Anomaly Detection. An anomaly detection architecture for the IIoT-based SM is proposed to deploy one of the top most concerned networking technique - a Federated Learning architecture - that can detect anomalies for time series data typically running inside an industrial system. The architecture achieves higher detection performance compared to the current detection solution for time series data. It also shows the feasibility and efficiency to be deployed on top of edge computing hardware of an IIoT-based SM that can save 35% of bandwidth consumed in the transmission link between the edge and the cloud. At the expense, the architecture needs to trade off with the computing resource consumed at edge devices for implementing the detection task. However, findings in maximal CPU usage of 85% and average Memory usage of 37% make this architecture totally realizable in an IIoT-based SM.

Keywords: Smart Manufacturing, Industrial Control Systems, Anomaly Detection, Time series, Federated Learning, IIoT.

1. Introduction

With the growth of the volume of data collected in manufacturing, Big Data offers a tremendous opportunity in the transformation of today's manufacturing paradigm to smart manufacturing and helps us to have AI-driven IIoT solutions working in real-time and being more accurate and efficient. A smart manufacturing system is monitored by sensors, controlling operations based on intelligent computing technologies to improve product quality, system performance as well as minimizes costs [1, 2]. Such modern industrial control systems (ICS) are

critical to the operation of the national facilities such as natural gas pipelines, or power grids.

With the application of leading technologies such as the Internet of Things, Cloud Computing, and Artificial Intelligence to smart manufacturing, a modern ICS outweighs traditional manufacturing [3]. For example, ICSs issue commands to open/close hydraulic valves, turn on/off power switches, etc. Therefore, any misoperation in an ICS may lead to fatal financial loss, or environmental destruction.

However, the exponential growth of IIoT brings not only tremendous benefits but also significant challenges to designing and implementing ICSs related to the cyber-security problems. In fact, a modern ICS is not an isolated system anymore, but is connected to the Internet. Hence, it would result in severe and heavy consequences if hackers could gain access to control the network and steal the security-critical data, or malware and worms could invade and destroy the operating system of a factory. The IIoT-based Industrial Control Systems are now one of the top industries targeted by a variety of attacks.

Many real reported attacks against SM systems have been provided in [4]. Therefore, the problem of protecting IIoT systems against cyber-attacks is becoming increasingly important and indispensable in their design. Various techniques such as firewall, antivirus or Intrusion Detection Systems (IDS) have been proposed. However, as threats become increasingly more complicated, there is a need for an anomaly detection algorithm that can discover attacks timely and accurately while still being lightweight enough to be deployed in IoT devices with limited computing powers in industrial settings.

From another perspective, in a Smart Manufacturing environment, centralized cloud computing of all data collected in manufacturing is deployed to empower the workloads and applications, reduce costs, and increase release velocity and agility. However, a SM always requires massive analysis in real-time, so that offloading computationally intensive tasks to a cloud centre may result in a delay, due to the time needed to transmit, process, and receive a large amount of data. To overcome this limitation, the concept of Edge computing came into

play in a smart factory [5]. This distributed approach can solve the Big Data processing issue in SM where data is collected from various sensors such as pressure, flow, speed.., so as generating a huge volume of data inside smart factories. Edge computing can quickly perform a necessary task in the network edge, i.e., between data sources and the cloud centre. Hence, the workload concentrated in the central cloud can be reduced.

Therefore, in this research, we propose an IIoT decentralised architecture for detecting Anomaly in Industrial Control Systems (ICS), where the processing intelligence is performed near to the data sources. In this edge-cloud architecture, we propose to:

- Decentralize the anomaly detection task to the edge where a hybrid model of Variational Autoencoder (VAE) [6] and Long-Short Term Memory (LSTM) [7] is deployed to cope with anomaly detection for time-series data. The hybrid model is suitable for detecting anomalies over multiple time scales. The hybrid model is designed with an optimized threshold using Kernel Quantile Estimator (KQE) to have high detection accuracy. In addition, the VAE module inside the hybrid model is also designed with only 2 fully connected layers at the VAE encoder and decoder to get high accuracy and compute fast enough so that suitable for being deployed in a low-computing capacity hardware like an edge, respectively.

- Develop Federated Learning (FL) to only send the training model of each edge to the cloud for the global update. This way helps to reduce bandwidth occupied in the link between edges and the cloud. FL supports the detection model right at the edge to have a faster system response upon attack arrivals, but providing a platform to have a global update on the training model of each edge that monitors a singular area.

The overall FL architecture is proved to operate efficiently in terms of CPU and Memory usage, Power consumption at edge hardware. In our case study, designing and implementing the training/anomaly detection task right at an edge device will take up to 85% of CPU usage of such embedded hardware

in the worst case; and 37% of Memory usage of the edge on average. These indicators show that the proposed architecture can be totally implementable in an IIoT industrial system with deployment of edge computing.

Besides, it also reduces bandwidth consumed in the transmission link between the edge and the cloud by 35%. It means, we can save 35% of the bandwidth for other application traffic which runs on top of the industrial system.

In the centralized learning manner, our detection algorithm with the optimization of the threshold is proved to outperform the same VAE-LTSM algorithm with heuristically-found threshold [8] in terms of precision, recall, F1-score.

In the FL mode, the hybrid algorithms still achieves good detection performance which is quite similar to the centralized learning mode where all raw data is sent to the cloud centre for training. And it is a very promising indicator for efficient Edge-Cloud computing where the detection accuracy is high while detection task is offloaded from the cloud to the edge which is nearer the data sources. Hence obviously the system can respond to cyber attacks faster.

2. Related Work

2.1. Cyberattack Detection for Industrial Control System

Recently, there have been a number of researches in the field of detecting cyberattacks for Industrial Control System [9, 10, 11, 12, 13, 14, 15]. The authors in work [9] propose a process mining method based on the log data of the devices in the system to make anomaly detection more efficient thanks to the rich source of information of the logs. In work [10], the authors investigate and use the one-class classification algorithms SVDD and KPCA for intrusion detection in the SCADA system. The computational cost when implementing these solutions in practice is always an important issue in the solution development process. To solve this, in work [11, 12, 15], the authors propose combining the input dimension reduction technique with the attack detection algorithm. However, in these studies, the authors have not evaluated and considered the problem of

live detection, when the task of detecting attacks in industrial control systems requires fast, accurate, and real-time detection. They also have yet to develop a distributed deployment strategy that responds to the scale of large production systems in real production.

Cloud computing has previously been applied to manufacturing control, such as in the case of software-defined networking-based manufacturing. The authors in [16, 17] build a SDN-based architecture to solve related cyber-security issues in IIoT and Industrial Control Systems. With the separation of a traditional network into the data plane and the control plane, SDN offers many benefits in monitoring and detecting network attack problems such as directing traffic when the attack occurs [16]. In these studies, the centrality of SDN is demonstrated by the fact that all decisions are centrally handled at the controller. This leads to system response latency issues as well as placing the processing burden on the controller. Given the distributed scale of manufacturing systems, these disadvantages of SDN are a serious problem. We seek to solve these limitations with our proposal of a distributed architecture with the detection model at the edge.

2.2. Anomaly detection for time-series data

In the field of anomaly detection mechanisms for time-series data, there are many research studies up to now [18, 19, 20, 21, 22, 23, 24]. In these studies, the anomaly detection at different times of the system is performed by the LSTM algorithm [18, 24]. In addition to being effective in detecting anomalies, Autoencoder is a technique used alone or in combination with LSTM in these studies [21, 23].

The deep generative network is a technique widely applied in solving anomaly detection problems. In particular, networks are composed of symmetric encoders [21, 23]. In work [23], the authors propose a deep architecture based on VAE to detect a cyber-attack on the water distribution system. VAE's symmetric architecture is used to reproduce the input through encoders and decoders to extract information for anomaly detection. However, extracting the sequential

information of the time series data is still a limitation of VAE. The best overall score in this paper is at 80%, and only 62.4% for the classification task.

In the same direction, the authors in [18, 21] also use a deep network model with an asymmetrical structure. In that study, a robust time-series anomaly detection framework is proposed based on a convolutional neural network with a structure of two symmetrical encoders and decoders. The decomposition in this architecture has an important circular role in handling patterns in the time series data for efficient detection. To serve the cloud and IoT monitoring, the authors in work [21] evaluate QoS parameters and consider the problem of processing centralized data in the cloud that requires a cloud infrastructure with a large storage capacity and strong processing capacity. Nevertheless, using the convolutional structure (i.e convolutional LSTM [18], convolutional Encoder-Decoder [21]), is time-consuming as well as requires a large amount of data to be effective for the training process. This may not be afforded by the resource limitation of edge devices. Research [25] has shown that with a model based on a neural network structure, it is not necessary to have a structure that is too complex to be effective in classification or detection.

Therefore, in this study, we use fully-connected to build the encoder and decoder structure instead of using multi-convolutional classes to ensure the performance of the algorithm in extracting information from time-series data.

Combining unsupervised anomaly detection and prediction model under one framework can solve the problem of requiring labels of data in anomaly detection methods as well as ensuring accurate prediction for time series data, as elaborated in [20, 22, 26]. In work [20], the authors propose an un-supervised real-time anomaly detection algorithm to execute anomaly detection in Smart Manufacturing with data collected from sensors in the factory production line. The model combining AE and LSTM can early and accurately detect defective products to minimize cost and time. In fact, it is difficult to determine the distribution of real time-series data because it is continuous series over time. So using VAE instead of AE to re-represent the extracted data in a distribution yields more principled and objective probabilities. From another point, in this

method we need a threshold to distinguish anomalous events.

A similar method is presented in work [8]. By a heuristic way, the authors use the construction error from the training set to compute the threshold by testing different values and choosing the threshold value that gives the highest detection performance. The threshold is selected via the trade-off among the detection performance measures such as precision, recall, and F1-score. This method is highly data-dependent and random.

To improve this, we use the Kernel Quantile Estimator theory to be able to determine the most appropriate threshold, suitably adaptive to the distribution of each different training data set.

2.3. Federated Learning

From the Federated Learning perspective, work [27] concerns Federated Learning due to its distributed and privacy-preserving nature, which is commonly deployed in edge computing as well as industrial IoT settings for anomaly detection. In this area, work [28] proposes a Federated deep reinforcement Learning empowered Anomaly Detection (FLAD) in order to detect anomalous users trying to leak private data in large-scale industrial settings. The paper proposes using Anomaly Detection Centers (ADCs) divided into three levels: Global, Local and Regional. With the Global ADC as the aggregator, the Regional ADCs will learn a detection model using FLAD, with which they will use to detect anomalous users. However, there is a chance that some Regional ADCs are compromised, so there will also be a federated model in the Local ADCs to detect them.

Rather than simple adoption, other papers seek to improve the FL framework in various ways. Study [29] concerns that some clients participating in FL could have bad data or be under attack, and thus will send low-quality weights to the aggregation server that negatively affects the global model. It proposes the server to test each local model sent by each client on some preset data first, and discarding those that yield a too high loss value. Work [24] and [30] meanwhile try to optimize the communication cost. Work [24] proposes compressing the

gradients by only transmitting local gradients that are larger than a certain threshold, while research [30] proposes putting edge servers between the clients and the central cloud. Some gradients calculation will be offloaded to these edge servers, and they will also communicate with the cloud for FL instead of the clients. There is controversy that the edge servers, being closer to the clients, will have higher bandwidth, and also the local model aggregation in the edge servers will lower the number of global communication rounds with the cloud required. These modifications require a significant architectural change or extra computational load. Therefore, in this paper, we opted for the solution of using a more simple detection model instead.

3. Federated Learning-based Detection System

3.1. Federated Learning-based Architecture for Smart Manufacturing

In this study, we propose a Smart Manufacturing architecture using FL for anomaly detection. By taking advantage of Edge Computing, the anomaly detection function is implemented on an Edge device with reasonable processing capacity instead of being carried out in the cloud server as usually seen in traditional architecture. It acts as the central monitoring and processing unit of a local and small area to detect anomalies.

Moreover, FL techniques allow the information between different localities to be shared through a centralized aggregator in the cloud, to serve the anomaly detection process at each edge more accurately without gathering all data from different edges to perform training.

As shown in Fig 1, our proposed architecture with three main components include:

- Factory sites: the different factories, or production lines that are divided into areas under the control of the device called the Edge server. In each of these areas is a system of sensors that collects the necessary data for the edge device to monitor anomalies.

- Edge device: devices with sufficient processing capacity corresponding to the size of each monitor area. It is a special feature in this federated-learning-based architecture where anomaly detection and monitor functions are located. Accordingly, this device receives data collected from sensors and performs the anomaly detection process.

- Cloud server: The global processing unit is responsible for collecting information from different Edge devices, then aggregating and creating a federated model, and updating the whole edge devices in the system.

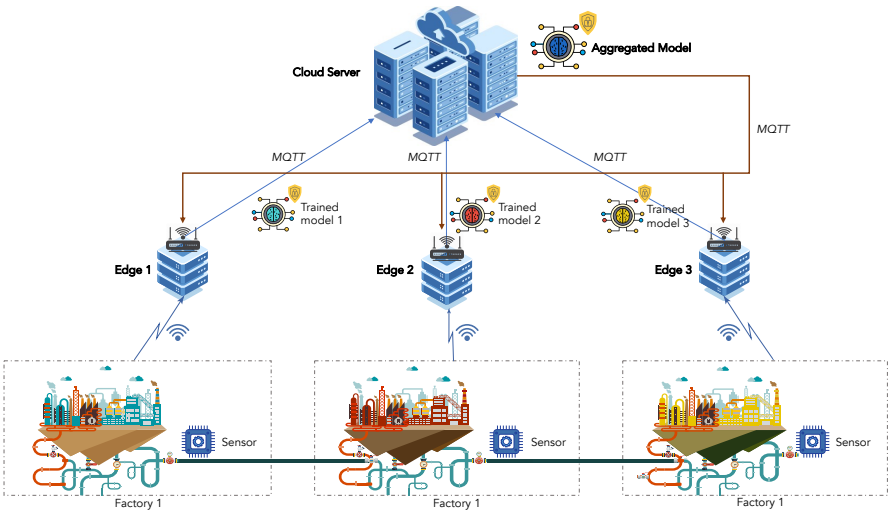

Figure 1: Federated-Learning-based Anomaly Detection Architecture for Smart Manufacturing

In this FL-based architecture, the training and detection process is performed at each edge device with local data of each manufacturing area and, the Edge device only sends information about the weight matrix of the trained model to the cloud server without having to send the whole raw data as a traditional cloud-based training system usually does. Although the cloud can have large storage and enough computing capacity to handle the volume of data collected in manufacturing, the computationally intensive tasks and the large data storage are located in the cloud servers may result in a delay. Due to the time

need to send, transmit and process a large amount of data from IoT devices in factory sites. This is a serious problem in a smart factory that must perform massive monitoring and detection in real-time. In this proposed architecture, the concept of Edge-Cloud Computing combining with FL can overcome this limitation.

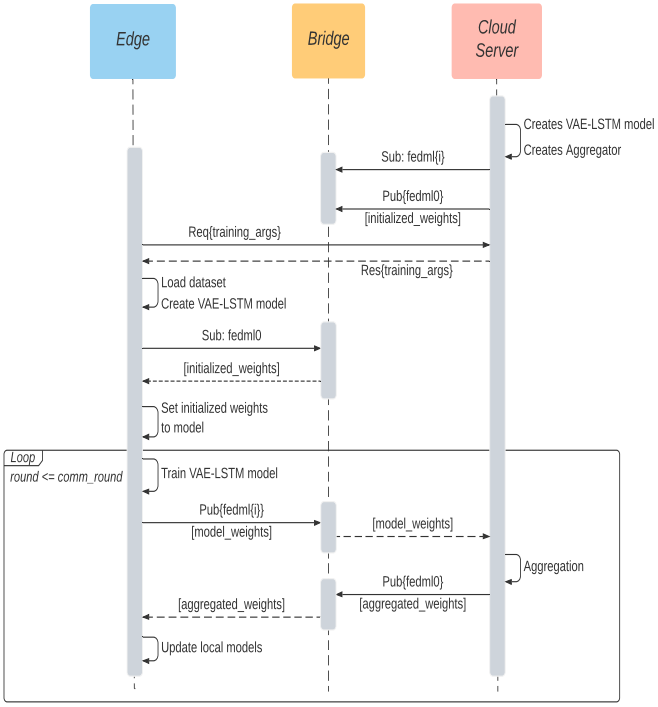

Figure 2: Flow chart of the FL-based Architecture operation

The operation of the FL mechanism in this architecture is illustrated in Fig 2.

- Cloud Server as a Weight Aggregator first creates the initial model.
- In this architecture, the hybrid VAE-LSTM model is used to overcome the anomaly detection with time-series data in smart manufacturing. It then subscribes to several MQTT topics to which clients will send their models' weights.

- After publishing the first model's weights to the aggregated model topics, the Cloud Server waits for requests of the specific VAE-LSTM model configuration from its clients.

- For the Edge side, with received configuration arguments, local models are created and set with initialized weights received by the subscriptions to the aggregated model topics.

- The Edge devices then train the local models with their own data sets. For every communication round, an Edge device sends trained models' weights w_{t+1}^k to Cloud Server for aggregation.

- The Cloud then computes the weight of the aggregated global model by the formula (1) introduced in [27]:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{1}$$

Where:

K:the number of participating clients

n_k: the amount of data that client k possesses

n: the total amount of data that all clients have

w_{t+1}^k: the weight of the local model at client k at time $t+1$

w_{t+1}: the weight of the aggregated global model at time $t+1$.

- This weight of the aggregated global model is finally sent downward to update the local model of each edge device.

3.2. Anomaly Detection using the hybrid VAE - LSTM model at the Edge

Anomaly detection in time series faces many challenges such as: contextual information, noise, concept drift,... Especially, anomaly detection in an Edge-computing-based IIoT environment has many additional requirements such as detection time and memory constraints [31]. To deal with this issue, the hybrid

VAE-LSTM model is developed at the edge devices where VAE stands for Variational Autoencoder and LSTM stands for Long Short-Term Memory Networks [7].

The idea of this hybrid model was actually initiated from research [8] for the fact that the model makes use of robust local features over short windows of VAE. Via VAE, the structural regularities of the time series are captured over local windows. It also utilizes the capacity of LSTM on estimating the long-term correlation in the time series over the features inferred by VAE. Hence, longer term trend can be modeled by LSTM. In fact, both VAE and LSTM are unsupervised learning, requiring no labelled data for anomaly training. Therefore, the hybrid model is an efficient detection method that can detect a new anomaly in a smart factory even it has not ever occurred before.

It is important to note that, in case the normal behaviour of IoT devices is changed over time, the performance of anomaly detection methods using one-class classification techniques may be affected by the increase of false alarms. To deal with this problem, we can use the idea of research [32] to develop a solution to correct the threshold. From the new normal data, the probability errors will be calculated using the trained hybrid VAE-LSTM model. It will then be combined with the probability errors of the original training dataset. Finally, the threshold will be recalculated using the KQE technique.

In this research, we provide a full description of the step-by-step deployment of the VAE and LSTM in a Federated-learning IIoT environment. The VAE model is designed with 2 fully connected layers in the encoder and decoder to be more light weight in terms of computing requirement. The VAE model then can be more suitable for deployment in such a hardware with low-computing capacity like an Edge. We also enhance the model by optimizing the threshold using Kernel Quantile Estimator (KQE) to have better detection performance instead of using a heuristic threshold as proposed in [8].

The full description of the general operation of the VAE-LSTM model is illustrated in Fig.3.

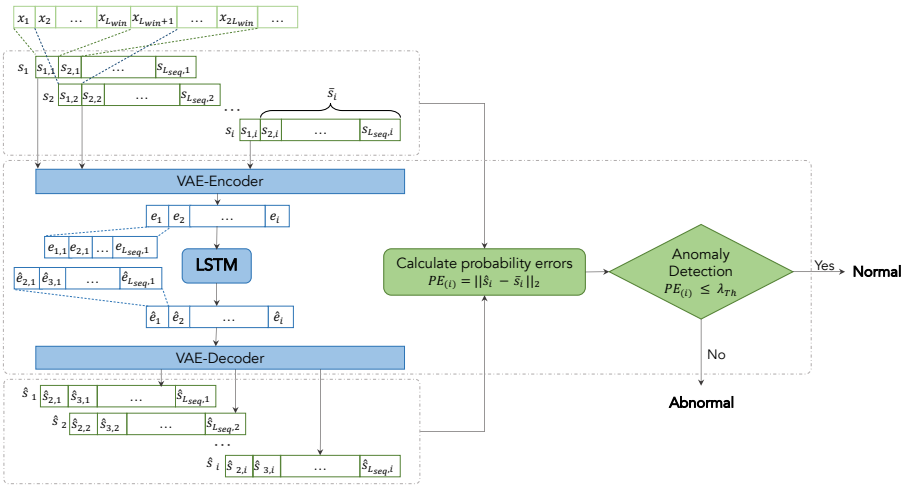

Figure 3: Block Diagram of the VAE-LSTM approach

- First, the time-series data $\{x_1, x_2, \ldots, x_{L_{win}}, x_{L_{win}+1}, \ldots\}$ - in which $x_i \in R^D$ is a multivariate data with D-dimensional - will be sliced into rolling windows of size L_{win}, which will be used to train the VAE algorithm.

- A sequence - s_i, is made by concatenating L_{seq} non-overlapping windows, which will be denoted as $\{s_{1,i}, s_{2,i}, \ldots, s_{L_{seq},i}\}$.

- After the VAE module has been trained, the sequences $\{s_1, s_2, \ldots, s_i\}$ are fed into the Encoder of the VAE module, where each window is compressed into a lower dimensional code. A code sequence e_i will consists of multiple codes - $\{e_{1,i}, e_{2,i}, \ldots, e_{L_{seq},i}\}$. The code sequences $\{e_1, e_2, \ldots, e_i\}$ will then be the input of the LSTM module.

- The LSTM module uses the first $L_{seq} - 1$ components from the current code sequence i ($\{e_{1,i}, e_{2,i}, \ldots, e_{L_{seq}-1,i}\}$) as input to return the prediction of the last $L_{seq} - 1$ components: $\hat{e}_i = \{\hat{e}_{2,i}, \hat{e}_{3,i}, \ldots, \hat{e}_{L_{seq},i}\}$

- Each predicted code sequence $\hat{e}_i$ is then recreated by the trained VAE decoder to become predicted sequence value $\hat{s}_i = \{\hat{s}_{2,i}, \hat{s}_{3,i}, \ldots, \hat{s}_{L_{seq},i}\}$.

- Comparing each predicted sequence $\hat{s}_i$ with $\bar{s}_i$, which is the input sequence

s_i without the first component, yields a series of probabilities errors, or also called prediction errors.

$$PE_i = ||\hat{s}_i - \bar{s}_i||_2, \tag{2}$$

- A threshold λ_{Th} is used on these errors to classify abnormal points, which is defined using a technique called Kernel Quantile Estimator (KQE).

Detailed background on LSTM and VAE operation as well as optimization of threshold λ_{Th} by Kernel Quantile Estimator (KQE) will be given in the following sections.

3.2.1. Deployment of Long Short-Term Memory Networks

LSTM (Long Short-Term Memory Networks) is utilized to act on the low-dimensional embeddings produced by VAE, to manage the sequential patterns over the longer term. Although there are many variations of LSTM, the comparison in [33] has shown that these variations are almost the same, some of which are more effective than the others but only in some specific scenes. Therefore, in this study, we use a simple LSTM Network with one cell to minimize the computing complexity of the system while ensuring the performance in processing time-series data. The basic structure of one cell in a LSTM network at state t is shown in Fig 4.

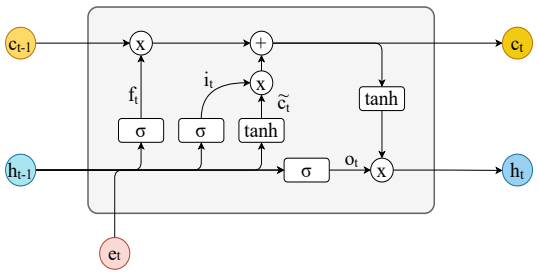

Figure 4: A Long-Short Term Memory cell

A LSTM cell has input e_t, output h_t (hidden state) and cell state c_t at time t; as well as c_{t-1} and h_{t-1} which are the output of the cell at the previous step.

Different to Recurrent Neural Network (RNN) cell which has only one *tanh* function, in order to remember the past data, a LSTM cell has three control stages:

- The first stage is the forget gate: With the input data e_t and previous output h_{t-1}, the gate will decide which information should be kept and which should be forgotten, that is performed by the function:

$$f_t = \sigma(W_f \times e_t + W_f \times h_{t-1} + b_f) \tag{3}$$

  Where:
  (W_f, b_f): the weight matrices and bias of the forget gate.

- The second step uses to process information to decide how much information should be add to the cell state c, called the input gate.

$$c_t = c_{t-1} \times \ f_t + i_t \times \tilde{c}_t \tag{4}$$

  In which i_t and $\tilde{c}_t$ are computed by:

$$i_t = \sigma(W_i \times e_t + W_i \times h_{t-1} + b_i) \tag{5}$$

$$\tilde{c}_t = tanh(W_c \times e_t + W_c \times h_{t-1} + b_c) \tag{6}$$

  In this step, the values i_t is computed by a sigmoid function along with a vector of values $\tilde{c}_t$ generated by a *tanh* function at the same time. (W_i, b_i) and (W_c, b_c) in (5), (6) are the weight matrices and the biases of the input gate and cell state, respectively.

- The last step is the output gate stage: The output h_t of LSTM depends on input e_t, h_{t-1} and the current cell state c_t. The *tanh* function helps to scale down the value's range to a number between -1 and 1, and the *sigmoid* function helps to filter the information from the cell state being output through the number within [0,1].

$$h_t = O_t \times tanh(c_t) \tag{7}$$

Which O_t is defined in (8) and (W_o, b_o) are the weight matrix and the bias of the output gate.

$$O_t = \sigma(W_o \times e_t + W_o \times h_{t-1}+) \tag{8}$$

In this structure, c_t is like a connection between the states with the current state of the network. This helps to keep the previous important information in the next states, providing long-term memory for the LSTM model.

3.2.2. Deployment of Variational Autoencoder

An autoencoder (AE) is a symmetrical, unsupervised neural network with the center hidden layer having fewer nodes than the input and output layer (a "bottleneck"). It is trained to recreate the output to be as close to the input as possible. After this process, the network has learned to compress the input to the bottleneck layer and then subsequently restore the input from it. This middle layer thus becomes the "latent representation" of the input, retaining most information about the input using fewer features. The part of the network before this layer becomes the encoder, and the part after becomes the decoder.

A variational autoencoder (VAE) [6] is a combination of the AE with the Variational Bayesian method. With VAE, the neural network describes probability distribution functions instead of deterministic ones. VAE is used to summarize the local information of a short window into a low-dimensional embeddings as shown in Fig. 5.

The purpose of VAE is to make sure that the latent space is regularized, and thus allows the generation of new, similar data. With the traditional AE, the model only cares about minimizing the reproductive loss and not at all about the state of the latent space of encoded data. Thus, the space will most likely be discontinuous and heavily fragmented, and decoding a random data point in latent space will probably result in nonsense data. On the other hand, the VAE will create continuously distributed encodings, and thus allows generation of new, similar data.

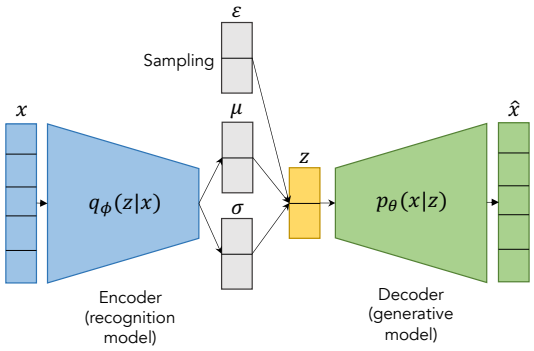

Figure 5: Illustration of a VAE Structure

Many different designs for VAE have been proposed, using various types of layers such as Dense, LSTM or CNN [34]. In this paper, we design the VAE encoder and decoder with only two fully connected layers each. This design is to achieve the simplicity and light weight of the model which is then capable of being trained on edge devices with limited hardware capacity; as well as reducing the communication cost, while still giving a satisfactory detection performance.

Let $\mathbf{x}$ be data generated by inputting a latent variable $\mathbf{z}$ through a random process with parameters θ. The posterior probability can be described as:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} \tag{9}$$

Because both the latent variable and the parameters of the process are not observable, the posterior probability as well as the marginal likelihood $p_\theta(x)$ are often intractable. To solve this problem, a recognition model $q_\phi(z|x)$, which is an approximation of the posterior, is introduced. The variational parameters ϕ of this model will be optimized so that:

$$q_\phi(z|x) \approx p_\theta(z|x) \tag{10}$$

The marginal log likelihood of the data can be written as:

$$\log p_\theta(x) = D_{KL}(q_\phi(z|x)||p_\theta(z|x)) + \mathcal{L}_{\theta,\phi}(x) \tag{11}$$

with $\mathcal{L}_{\theta,\phi}(x)$ being the *evidence lower bound* (ELBO). Because the first term of Eq. (11) is the Kullback-Leibler divergence and is always non-negative, we

always have $\log p_\theta(x) \geqslant \mathcal{L}_{\theta,\phi}(x)$. Thus, optimizing the ELBO with respect to both θ and ϕ will minimize this KL term, bringing $q_\phi(z|x)$ closer to the true posterior. The ELBO is defined as:

$$\mathcal{L}_{\theta,\phi}(x) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}(\log p_\theta(x|z)) \tag{12}$$

The first term of Eq. (12) is the KL divergence of the distribution of the encoding with respect to the normal distribution, to ensure that the latent space is continuous. The second term is the reconstruction loss to make sure that the data is being encoded correctly.

A common choice is to have both $q_\phi(z|x)$ and the prior $p_\theta(z)$ be Gaussian. The ELBO to be optimized for each data point i can then be given as:

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(x^{(i)}) = \sum_{j=1}^{J} \frac{1}{2}\left[1 + \log((\sigma_j^{(i)})^2) - (\sigma_j^{(i)})^2 + (\mu_j^{(i)})^2\right] \\ -\frac{1}{L}\sum_{l=1}^{L} \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x^{(i)}|z^{(i,l)})\right]\end{aligned} \tag{13}$$

Where μ and σ are the output mean and standard deviation vectors of the encoder, J is the size of those vectors, and L is the number of times a sample is taken from the latent space.

The parameters of the model can thus be expressed as:

$$(\theta^*, \phi^*) = argmax_{\theta,\phi}\mathcal{L}_{(\theta,\phi)}(x) \tag{14}$$

Instead of maximizing $\mathcal{L}$, we can simply take its negative as a loss function to minimize, then apply gradient descent to find the optimal parameters. However, calculating the gradient is unstable due to the sampling of the random variable z in the second term of Eq. (13). To circumvent this, a technique called reparameterization [6] can be used, in which instead of sampling the latent variable z directly, we write $z = \mu + \sigma \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$ and sample ϵ instead. That way the random sampling is externalized and the gradient descent can be calculated as usual through deterministic nodes.

3.2.3. Threshold optimization using Kernel Quantile Estimator

. As aforementioned in Fig. 3, the anomaly detection is based on a set of prediction scores, and we need to set a threshold for this set of scores to defining what is anomaly or normal. Usually, this threshold can be determined by several rules such as 3σ and 2σ by assuming data has a normal distribution. As in [35], the author uses the maximum likelihood estimation theory to determine the threshold value for abnormal points detection with the assumption that the "error vector" follows by a multivariate Gaussian distribution.

In fact, in most contexts, it seems difficult to confirm that realistic data have a normal distribution. To solve this problem, we can determine the threshold independently from the data distribution by applying a non-parametric method. In this study, we use the Kernel Quantile Estimator (KQE) method [36] to estimate the λ_{Th} for the training data set based on probability error follow by (2).

In (2), we have a set of prediction scores $Score = \{PE_1, PE_2, \ldots, PE_k\}$.

Suppose that $PE_1 \leq PE_2 \leq \ldots \leq PE_k$ are reorganized corresponding to the order statistics of the reconstruction error. KQE is used to calculate the λ_{Th} as shown in Equation. (15):

$$\lambda_{Th} = \sum_{i=1}^{N} \left[\int_{\frac{i-1}{k}}^{\frac{i}{k}} K_h(t-p)dt \right] PE_{(i)} \tag{15}$$

Where K is the density function symmetric about zero; bandwidth h (greater than 0) is used to control the smoothness of the estimator for k samples in the prediction score set, and p $(0 < p < 1)$ is the preset value.

In equation (15), the selection of kernel for K and bandwidth for kernel density estimator should be in focus. The author in [36] has shown that the estimation performance does not have many effects by the choice of function K. In this study, we use the standard Gaussian kernel for K like in (16).

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{z^2}{2}) \tag{16}$$

In contrast to K, the smoothness of the density estimate is affected significantly

by bandwidth h. To overcome this problem, an asymptotically optimized bandwidth like in [36] is computed as:

$$h_{opt} = \sqrt{\frac{p(1-p)}{k+1}} \tag{17}$$

Threshold λ_{Th} is used to determine the anomalous points in the time series data. By using the Kernel Quantile Estimator method, we can solve the unknown distribution of the data, ensuring the accuracy of anomaly detection.

4. System performance evaluation

In this section, we evaluate the performance of our proposed overall architecture in terms of anomaly detection performance and effectiveness of the Federated-learning-based Edge computing. The main case study is smart manufacturing in a Gas Pipeline Factory with a realistic time-series data set achieved from the Supervisory Control and Data Acquisition (SCADA) systems [37].

However, in order to assess the anomaly detection performance of our solution in various industrial contexts, we also use other time-series data sets collected in several different fields such as ECGs [38], respiration [38], power demand [38], gesture [38], space shuttle [38] and NYC taxi [39]. The short introduction of these data set are shown in the Table 1.

4.1. Data set pre-processing

Data sets are the input for any training model developed at either the Edge or the Cloud. However, those data sets frequently have many missing feature values, so it is always necessary to clean and pre-process data before the training process. For example, as our main case study, the SCADA data set is shown in Fig 6 with many missing values (i.e "?" values). The SCADA data set includes 274,628 samples and, each sample contains network information, payload information etc.

To deal with the missing values, we can crop the samples or features that include those missing values. However, those cropped samples and features may

Table 1: List of the data sets used in our experiments

Data sets	Description
Gas pipeline (SCADA) [37]	The gas pipeline data set was collected in 2015 by Mississippi State University Lab for anomaly detection using their in-house gas pipeline system
ECGs [38]	The data set on Electrocardiograms is a time-series of the electrical signals caused by heartbeats
Respiration [38]	The respiration data set contains information on Patient's respiration measured by thorax extension when waking up
Power demand [38]	The data set provides information on the power consumption of a Dutch research facility in 1997
Gesture [38]	The gesture data set consists of 2 features representing the coordinates of the actor's right hand while performing a variety of actions
Space shuttle [38]	The data set measures the solenoid current of a Marotta MPV-41 series valve cycled on and off
NYC taxi [39]	The data set contains information on The New York taxi passenger data stream from July 2014 to June 2015

Time	Address	Function	Length	Setpoint	Gain	Reset rate	Deadband
1418682163	4	3	16	?	?	?	?
1418682163	4	3	46	?	?	?	?
1418682165	4	16	90	10	115	0.2	0.5
1418682165	4	16	16	?	?	?	?
1418682167	4	3	16	?	?	?	?
1418682167	4	3	46	?	?	?	?
1418682169	4	16	90	10	115	0.2	0.5
1418682169	4	16	16	?	?	?	?

Figure 6: Examples of the missing values in the SCADA data set.

contain useful information for precise training. Therefore, carelessly removing them can lower the model's precision.

According to work [40], there are 3 types of missing data mechanisms: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). MCAR means the missingness of data is unrelated to any values. MAR indicates that the tendency of a value to be missing might depend on the observed data, but not the missing data. In contrast, MNAR indicates that there is a relationship between the missingness and its value. Considering the missing values as shown in Fig. 6, we notice that the SCADA data set is likely to have the characteristic of MAR. In the same direction, the authors in work [41] also base on the LOCF method to process missing values in the SCADA data set, the results show that this method is more suitable and effective than some other methods in the anomaly detection problem. Therefore, we apply Last Observation Carried Forward (LOCF)[42], a well-known method to handle MAR, to process the missing values in the SCADA dataset. LOCF uses the immediately preceding value of the same feature to fill in the missing value. If the data set begins with missing values, we use the first observed value to substitute them.

After handling the missing values, to normalize all features to the same scale and arithmetic values, we perform data transformation by the mean-standard deviation scale techniques as in (18) .

$$x'_i = \frac{x_i - \mu}{\sigma} \tag{18}$$

In which, μ and σ are the mean and the standard deviation of the listed feature values, respectively. To train and evaluate the performance of our solution in different scenarios, we divide the SCADA data set into the training set (160,870 samples) and testing set (68,657 samples) while the training set contains only normal samples for anomaly detection purposes.

4.2. Detection Performance Evaluation

In this section, we evaluate the performance of our approach in various contexts. The assessments were carried on the aforementioned data sets in terms of precision, recall, and F1-score. These metrics are defined by as follows:

$$Precision = \frac{true\_positive}{true\_positive + false\_positive}$$
$$Recall = \frac{true\_positive}{true\_positive + false\_negative}$$
$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where:

- $true\_positive$: an outcome where the model correctly predicts the positive class
- $false\_positive$: an outcome where the model incorrectly predicts the positive class
- $false\_negative$: an outcome where the model incorrectly predicts the negative class.

Note for time series data, an anomaly only occurs in one single time point, but is detected using a window, so in the ground truth array, all the points in this window are considered as anomalies. The algorithm may not recognize all, but only some elements in that window as outliers. Realistically, this means the true anomaly has been detected, but due to the way the ground truth is constructed, the algorithm is still being fined for false negatives. We resolve this with the same approach in [43], in which if any elements in a window are detected, the whole window is also considered detected.

4.2.1. Experiment 1

Experiment 1 is presented in Table.2 in which the performance of our Federated-learning detection model is compared with the cloud-centralized-based VAE-LSTM detection model with the heuristic threshold λ_{Th} mentioned in [8]. The

detection performance of the two approaches is compared on top of 7 different data sets. This comparison is to evaluate how close the detection performance of the FL approach can get to the current centralized approach with the same detection method.

In the centralized learning experiment, to achieve the best threshold by the heuristic way proposed by work [8], we tested with 25 different thresholds evenly spaced between the smallest and largest reconstruction error, and finally finding the best heuristic threshold for each of the 7 data sets. In addition, a notice that work [8] did not mention how the authors realized VAE, since there are different types of layers: Dense, multi-layer CNN... However, as we experimented, application of VAE with multi-layer CNN is implementable at the edge device since its computing complexity is so high and training time is so long even running the algorithm over a high-computing server, much worse for a low-computing-capacity edge device. Therefore, although the centralized solution with multi-layer-CNN works quite perfectly in terms of detection performance as shown in the original work [8], we decided to implement VAE of work [8] with Dense to reduce the number of layers, features, thereby decreasing computing complexity and the matrix of weight at the edge. This experiment and implementation make work [8] implementable at the edge device for the comparison, but leading to the different detection performance shown in this study compared to the original work [8].

In the Federated-learning experiment, the proposed threshold optimization using KQE is achieved by varying 9 values of p ranging from 0.1 to 0.9, with the distance of 0.1, in addition to 2 more values of 0.95 and 0.99 for a total of 11 values.

As shown in Table 2, the hybrid VAE-LSTM module performs well on many time-series data sets, but did poorly on some, such as the respiration or the gesture data set. We believe this could be improved by making the detection model more complex, but that would come at the cost of increased computation and bandwidth demand at edge devices so that the edge is not be able to handle.

But the VAE-LSTM proves to work very well in the the main case study - the SCADA data set in which Precision, Recall and F1-score get very high.

Furthermore, despite the fact that the Centralized Learning model tends to perform better than their Federated-Learning counterparts, and that the KQE approach only has less than half as many tries compared to the heuristic approach. However, surprisingly, our Federated-learning approach still outperforms the centralized learning VAE-LSTM solution proposed by [8] in most of the data sets. The only data set where our proposal performs worse on all metrics is stdb_308_0. This shows that a suitable threshold found by the KQE method can vastly improve the performance of an anomaly detection model, and that our KQE is better and faster at finding the optimal threshold in most cases.

Table 2: Federated Learning approach vs. Centralized Learning approach over 7 different time-series data sets

Data set	**Test set**	**Cloud-Centralized-VAE-LSTM [8]**			**Our FL-based approach**		
		Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Space shuttle	TEK14	0.5792	0.9990	0.7333	0.8623	0.8431	0.8536
	TEK16	0.9636	0.8881	0.9243	1	1	1
	TEK17	0.8961	0.9637	0.9287	0.9650	1	0.9822
Respiration	nprs43	0.6586	0.4952	0.5653	0.9313	0.5530	0.6939
	nprs44	0.9786	0.2799	0.4353	0.5347	0.5027	0.5182
Gesture	gesture	0.3422	0.9989	0.5098	0.5278	1	0.6910
Nyc taxi	nyc_taxi	0.7711	0.7628	0.7669	0.9606	1	0.9799
ECG	Chfdb_chf01_275	1	1	1	0.9175	1	0.9570
	chfdb_chf13_45590	1	1	1	0.9489	1	0.9738
	chfdbf15	0.6484	0.8968	0.7526	0.9458	1	0.9721
	ltstdb_20221_43	0.9607	1	0.9800	1	1	1
	ltstdb_20321_240	1	1	1	1	1	1
	mitdb__100_180	0.9754	1	0.9876	1	1	1
	qtdbsel102	0.5827	1	0.7364	0.9604	1	0.9797
	stdb_308_0	0.7521	1	0.8585	0.6073	0.6373	0.6220
	xmitdb_x108_0	0.6727	1	0.8043	1	0.7628	0.8654
Power demand	Power_demand	0.2728	0.8948	0.4182	0.7355	0.9100	0.8135
SCADA	Scada	0.9315	1	0.9645	0.9609	0.9982	0.9792

4.2.2. Experiment 2

In order to test the effect of the FL approach and KQE-based threshold optimization on the detection performance separately, we tested the heuristic

and KQE approaches for finding the best threshold on top of both centralized (e.g. the approach of work [8]) and FL scenarios, in the main case study - the SCADA data set. The results are demonstrated in Table 3.

We can see that in both of the centralized and federated case, the KQE-based scheme gives the better F1 score and Precision, and only has a slight reduction in Recall in the Federated-learning approach. With the KQE-based threshold selection method, the cloud-based centralized method proposed by work [8] slightly increases its own detection performance compared to the results shown in Table 2.

The best p found for the SCADA data set is 0.9. We also tested the AUC (Area under the ROC Curve) value for both scenarios, as this metric is not affected by the choice of threshold. We found that the Federated-Learning model only has a slightly lower AUC value than the Centralized-Learning model, proving that the decrease in model quality by FL is only minuscule.

Table 3: SCADA Centralized vs. Federated Results

Learning Approach	Heuristic-based threshold			KQE-based threshold				AUC
	Precision	Recall	F1	Best p	Precision	Recall	F1	
Centralized	0.9315	1	0.9645	0.9	0.9585	1	0.9788	0.8539
Federated	0.9315	1	0.9645	0.9	0.9609	0.9982	0.9702	0.8500

4.3. Evaluation on Edge Computing Efficiency

In this section, we desire to study the performance of a real edge hardware working in the Federated-learning IoT environment. This issue is important since we can estimate how well such a proposed Edge-computing architecture can work in reality with limited hardware capacity like an Edge.

Communication efficiency of the FL approach is compared with that of the Centralized Learning architecture. In addition, the aspects of computational resources and energy consumption at every single Edge are also covered.

At first, the detail of the devices, framework and measurement method used in the testbed is elaborated in the following section.

4.3.1. *Experimental setup and Measurement implementation*

Testbed setup

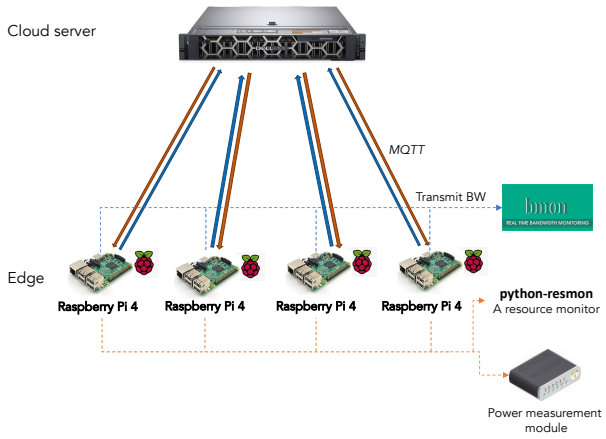

Figure 7: Testbed Setup

As illustrated in Fig. 7, our testbed involves:

- Four Raspberry-Pi-4-Model-B kits acting as edge devices; Raspberry-Pi-4 features a 1.5GHz ARM Quad-core Cortex-A72 processor and 4GB RAM with Raspbian OS 32-bit)
- A Dell Precision 3640 Tower workstation serving as a Cloud Server; the workstation features an Intel Core i7–10700K 3.8GHz (up to 5.1GHz), RAM of 16GB, Arch Linux.
- All Edge devices and Cloud Server are interconnected with a router via their wifi interfaces

In the edge devices, we implement our FL-based VAE-LSTM framework written in Python 3 with the Tensorflow 2 platform, which is built with the support of the FL framework - FedML. [44].

In our architecture, the edge devices and cloud server interchange weights and bias matrices of the VAE-LSTM model using the MQTT protocol which

is standardized for IoT environment. For better performance in the long run, *EMQ X Broker* [45] is hosted on the cloud server as a MQTT broker. *EMQ X Broker* is found an open-source broker that can be most scalable to accept more edge devices connecting to the server.

Implementation of Bandwidth occupation measurement

To measure bandwidth occupation in the link between each edge and the cloud server, we use the tool *bmon* [46] on the up-streaming direction of the edge devices' Wifi interfaces to measure bandwidth usage. *bmon* [46] is a monitoring and debugging networking tool, especially used to capture statistics of a network.

Implementation of Computation resource and Energy consumption measurement

For exact assessment of computational resources such as CPU and Memory usage as well as energy consumption, external monitors and peripheral devices for Raspberry Pis should not be used since they will increase the CPU/Memory usage and Energy consumption whilst we only want to measure the impact of the Federated-learning based VAE-LSTM model onto the edge hardware only. Consequently, it is needed to establish an SSH connection to those Edge devices so as to conduct measurement experiments. To display the measurement results, all measured values are logged into separate text files and processed later to display into graphs.

These SSH connections are set up over the Ethernet interfaces of the Raspberry Pi kits, as a substitute for WIFI ones, via a second router; since the WIFI interfaces are used for sending data from the edge to the cloud. This setting is to avoid noise hence more accurate bandwidth occupation measurement can be acquired.

In order to assess the computing occupation level of the edge devices, we use the tool *resmon* [47] to measure CPU usage and memory usage during the training task of the FL-based VAE-LSTM model; in comparison with the cen-

tralized learning model in which all data coming to the edge are just forwarded to the cloud for centralized learning.

The measurement of energy consumption involves using *UM25C USB Tester* [48] to calculate the energy usage of Raspberry Pi.

4.3.2. Experimental Scenarios

To compare the communication efficiency of our Federated-Learning architecture with the centralized learning architecture using the same type of detection mechanism, two main scenarios are established.

In each scenario, we evaluate with the same detection model (i.e. VAE-LSTM) and the same data set (i.e. SCADA of 58MB). We focus on four key metrics to evaluate the performance of an edge:

- Bandwidth consumption in the link between the edge and the cloud
- Power consumption at an Edge during training
- CPU usage at an Edge during training
- Memory usage at an Edge during training

Scenario 1 is for the cloud-based centralized learning architecture in which all data from clients and sensors are sent to the cloud for centralized training and detecting anomalies. Hence, the Edge just forwards traffic directly to the cloud server.

Scenario 2 is for the FL architecture in which incoming traffic is captured by edge devices. Then the training process as well as anomaly detection are taken place right at the edge. Hence, the edge only sends its own model (i.e. matric of weights) to the cloud for global updating.

Scenario 3 is for investigating the behavior of the edge during the detection phase of FL. To set up the experiment, a time-series test set of 2000 samples from the SCADA dataset is utilized. This set of data is then fed into the detection module which will process one block of 1000 data points at a time. In the detection phase, the output of anomalous points is determined by the

comparison of the reconstruction error with a threshold that was previously defined using the KQE method

4.3.3. Performance Results and Analysis

Bandwidth Occupation during the training phase

Fig. 8 demonstrates the bandwidth occupation for Scenario 1 - the centralized learning architecture where data coming from sensors are all forwarded to the cloud by the edge devices during 45 seconds of the experiment. As it can be seen, during 45 seconds, bandwidth occupation stays at a rate of approximately 2000 KiB per second on average for most of the time and reaches the rate of around 2600 KiB per second at maximum. (Note: 1 KiB is denoted for 1024 Bytes in this study).

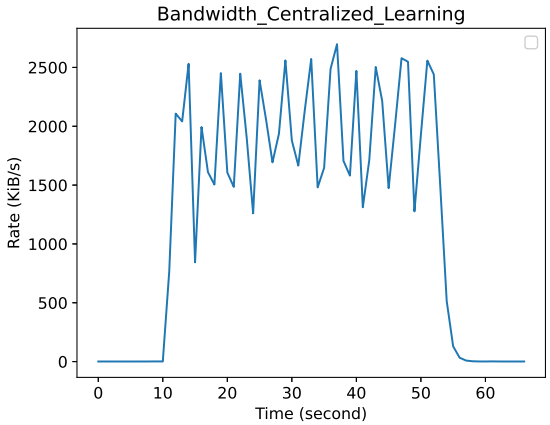

Figure 8: Bandwidth occupied in the Edge-Cloud link for the Centralized Learning approach

Meanwhile, for the FL architecture, since data coming from the sensors are processed for training right at the edge devices, thus, the edge just sends matrices of weights to the Cloud for further global updates. As Fig. 9 elaborates, during the whole VAE-LSTM training process of app. 1400 seconds, we almost get only one spike of bandwidth occupation of about 1700 KiBytes per second. And this spikes spans in only about 8 seconds as illustrated Fig. 10 as the zoom-in of Fig. 9.

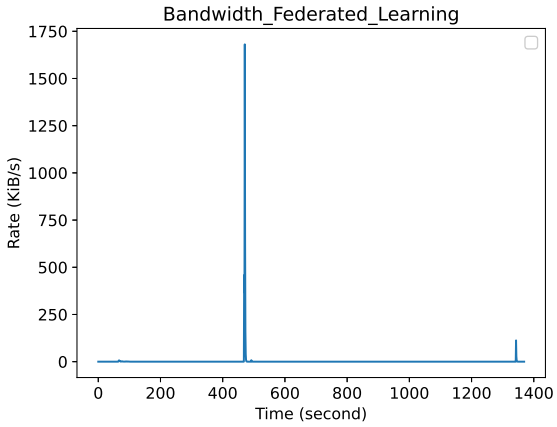

Figure 9: Bandwidth occupied during the VAE-LSTM training in the Edge-Cloud link for the Federated Learning approach

The results from Fig. 8 and Fig. 9 show that the proposed FL architecture saves bandwidth by 35% and takes only 18% of the the transmission time in compared with the centralized learning architecture. Therefore, deployment of such a FL architecture will leave the edge-cloud link to have more free link capacity to convey other information data of the Industrial IIoT system. Moreover, to mention again, such a FL model can help reducing the system response to attacks since the early attack prevention is implemented right at the Edge which is near the attack sources inside a smart factory.

In case, larger data size should be trained for the sake of detection performance, the FL approach will outperform the Centralized learning mode better in terms of Bandwidth occupation since the amount of model data (i.e. weights) transmitted to the Cloud remains the same while raw data from sensors transmitted to the Cloud for centralized training increases.

However, from another perspective, implementing a detection model distributively at every edge device may result in an trade-off with the resource and energy consumption at the edge. This issue is obvious since now all edge devices need to compute and process data by their own resource for the anomaly detection task. Therefore, it leads us to investigate the Power, CPU, and Memory

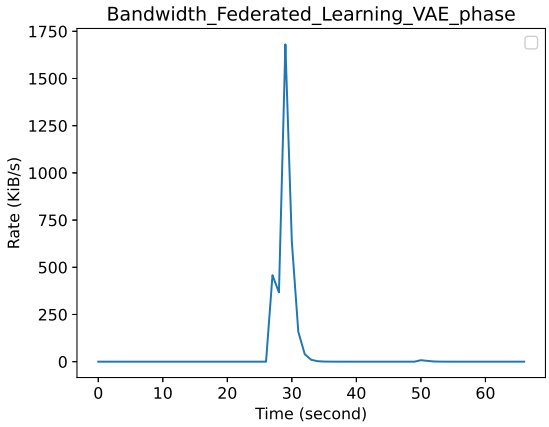

Figure 10: Zoom-in of the Bandwidth spike in Fig. 9

consumption of an edge during the training phase.

Power Consumption at the Edge during the training phase

In this experiment, power consumption in each single Edge device during the Centralized Learning process (i.e. forwarding-traffic process) and the FL process (i.e. training process) are measured.

As Fig. 11 illustrates, the power consumption at the edge in the centralized learning scenario is in the range of 2600-3600 mW. The baseline power consumption of the Raspberry-Pi-4 edge is presented in the first 20 seconds of Fig. 11, which stands for about 2600 mW. (Baseline power is defined as the state in which the edge is just switched ON without processing any task).

For the FL scenario, as Fig. 12 shows, the power consumed during the training process is divided into continuous phases: the VAE phase, the phase of producing embeddings for the LSTM phase and the LSTM phase.

For simplicity and display clarity of the Figure, we show the power consumption of the VAE-LSTM training process with only 1 communication round of the VAE phase, and 1 (embedding and LSTM) phase in order to demonstrate how much power an edge device (i.e. Raspberry Pi 4) consumes over time. As

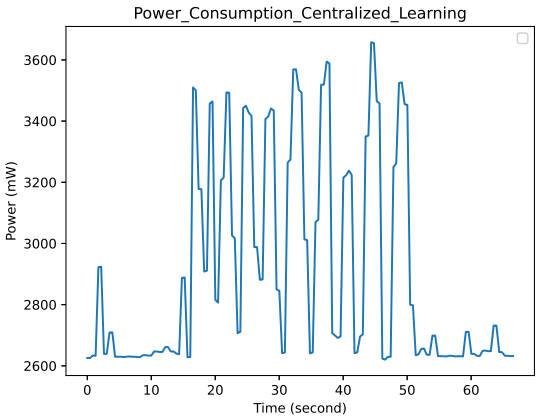

Figure 11: Power consumption at an Edge in Centralized Learning

measured in the Raspberry-Pi-4 edge device, for each round, it takes about 410 seconds for 1 round of the VAE phase, 850 seconds for producing embeddings and LSTM phase, and 1300 seconds to complete the whole training process.

In a real situation of running the VAE phase in 15 communication rounds to get high detection performance, the power consumption of the VAE phase illustrated Fig. 12 will replicate the same pattern of the first round 15 times over time.

And in order to complete the 15-round VAE-LSTM training process to get the high detection performance presented in Table. 3, we will have to run 15 times of the VAE phase and plus 1 embedding and LSTM phase. It will take an additional 410 seconds per VAE phase and take about 2 hours for the whole training process at the edge.

It is important to note that, in this research, the VAE-LSTM is used for feature extraction. In case there is a change in the network status but long training time is not allowed, we still can use the VAE-LSTM which was trained with the old data without retraining to reduce the training time. In this case, the new threshold recalculation will be based on both of the prediction scores computed from the old data and new data which is got through the trained VAE-

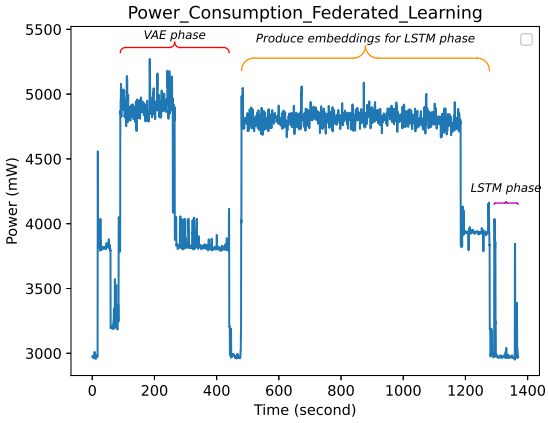

Figure 12: Power consumption at an Edge in Federated Learning

LSTM to provide corresponding prediction scores. Using VAE-LSTM only one time will reduce much of the training time in case long training time is not allowed to protect critical ICSs..

In conclusion, to develop such a proposed FL-based Edge-computing architecture, the system should be designed to monitor and detect anomalies inside an IIoT industrial system in every 2 hours. The edge device like Raspberry-Pi-4 must load average power consumption of 5000-6000mW.

CPU consumption at the Edge during the training phase

In the same experimental setup with the Power measurement, CPU usage in a single edge is also measured.

For centralized training scenario, Fig. 13 elaborates the CPU usage of an edge during 45 seconds of forwarding data from the sensors to the cloud.

For the FL scenario, Fig. 14 demonstrates the CPU usage in a single edge during the training VAE-LSTM process of 1 round of the VAE phase and 1 embeddings and LSTM phase.

As we can see, in the centralized learning architecture, since the detection task is offloaded onto the cloud server, the edge device is just in charge of

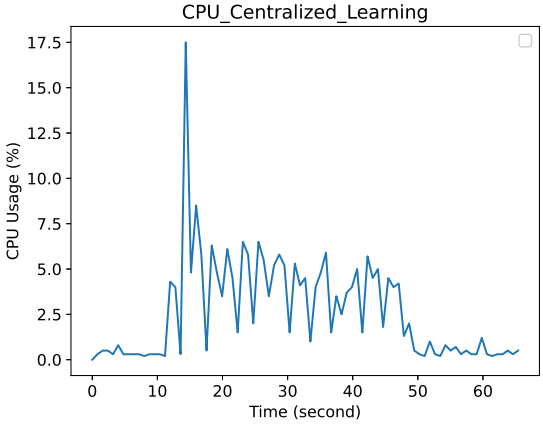

Figure 13: CPU Usage of an Edge in Centralized Learning

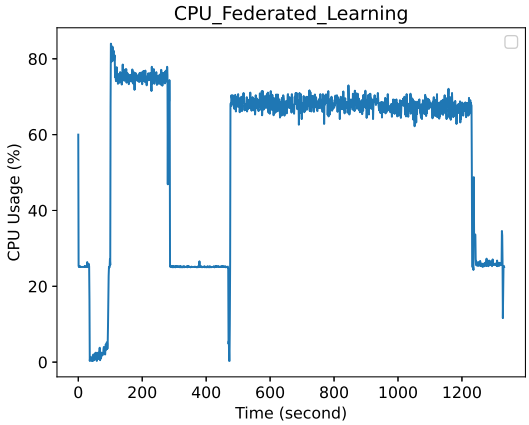

Figure 14: CPU usage of an Edge in Federated Learning

forwarding data from sensors, so the CPU usage reaches only 5% the total CPU on average. This figure of 5% also presents for the application traffic running on top of all hardware deployed inside a Smart factory like the Gas Pipeline factory.

Whilst, in the FL scenario, the training task takes 80% of the CPU in the worst case. However, the figure of 80% is a good figure since Edge's CPU still has 20% left for other application information which may account for another

5%. Therefore, the overall CPU usage for anomaly detection task and other traffic communication tasks account for roughly 85%. It shows that our proposed edge-computing architecture is definitely implementable in smart manufacturing.

Memory Usage at the Edge during the training phase

In the same experimental setting, we also present the memory usage of an Edge

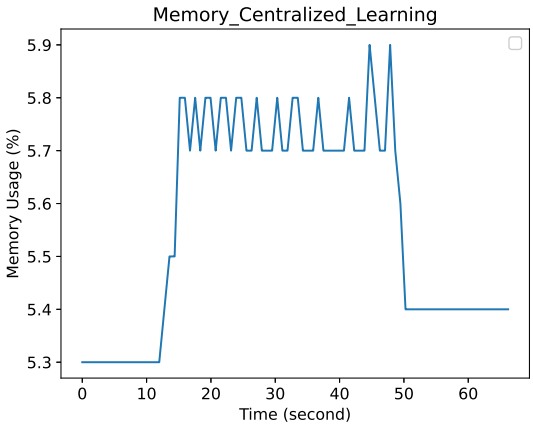

Figure 15: Memory Usage of an Edge in Centralized Learning

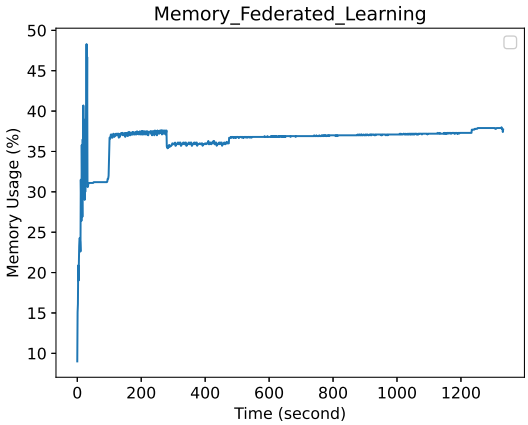

Figure 16: Memory Usage of an Edge for Federated Learning

for the Centralized Learning and Federated Learning scenario.

Fig. 15 shows the memory usage of an Edge in Centralized learning architecture which accounts for less than 6% at maximum over 45 seconds of the total experiment time. Whilst, in the FL architecture, the memory usage presented in Fig. 16 accounts for 37% on average during 1300 seconds of the training phase with 1 round of the VAE phase and 1 embedding and LSTM phase.

Power Consumption, CPU Usage, Memory Usage at the Edge during the detection phase

In Scenario 3, Power Consumption, CPU and Memory Usage of the Edge device during the anomaly detection phase are presented in Figure.17, Figure.18, and Figure.19 accordingly. As Figure.17 illustrates, the power consumption at the edge during the detection period mostly is in the range of 4000-4200mW in which note again that the baseline power consumption of the the Raspberry-Pi-4 edge accounts for 2600mW. The CPU usage during the detection phase takes about 60% on average, and memory usage takes about 11% in the worst case.

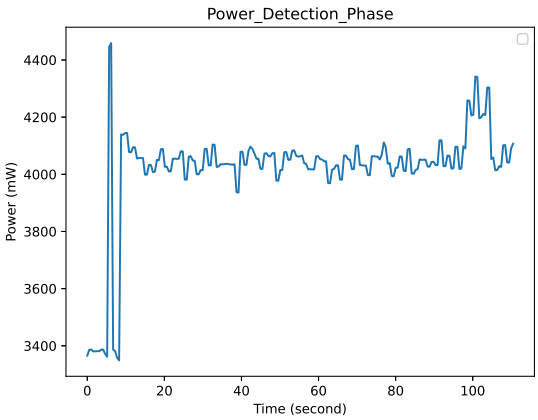

Figure 17: Power Consumption of an Edge device during the anomaly detection phase

Overall, this consumption is lower than that of the training phase which can

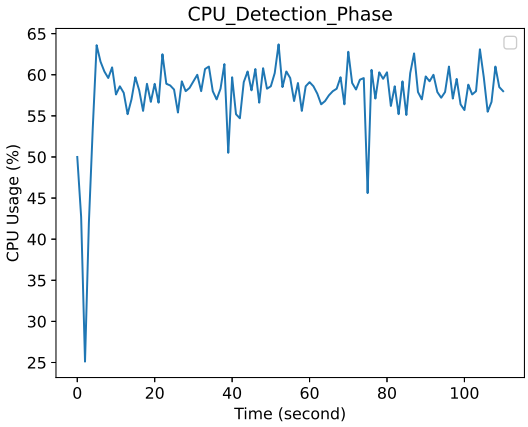

Figure 18: CPU Usage of an Edge device during the anomaly detection phase

be explained as the Edge device no longer has to deal with extensive mathematical computation relating to high dimensional matrices as the training dataset. Instead, much smaller chunks of data are fed as input to the detection module, hence leading to the significant reduction in power consumption as well as consumed computing resources.

5. Limitations, Discussion and Future Work

In this research, we have proposed a Federated-Learning architecture based on Edge-computing for smart applications of Smart manufacturing in the context of Big Data. The overall solution has shown to have high detection performance, whilst giving the advantage of having a fast response since anomaly detection is implemented near the attack sources (i.e the edge). The FL architecture distributes the monitoring and detection task to smaller local areas, so that it can deal with Big Data generated inside a smart factory better. In addition, the proposed system is proved to reduce bandwidth for controlling data in the transmission link between the edge and the cloud, meanwhile ensuring that the edge hardware will not be overloaded in terms of CPU and Memory usage.

For future work, we will try to optimize our system more in terms of weights

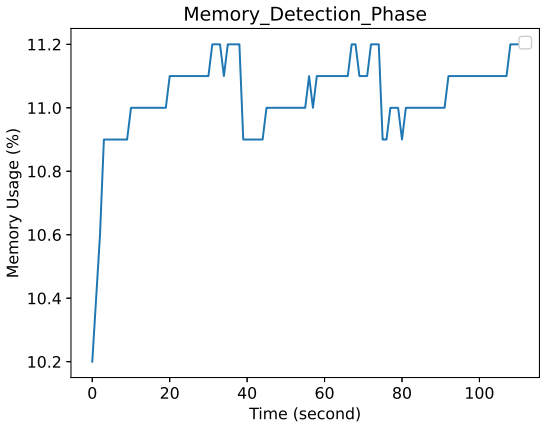

Figure 19: Memory Usage of an Edge device during the anomaly detection phase

communicating between the Edge and the Cloud, as well as defining the more suitable training data so that the Edge devices take up as little computing resource as possible to be able to carry on more other tasks. The limit of the data size and number of features should be studied to ensure the edge computing work in a stable mode. Moreover, the research can be extendable to other applications of SM such as: Predictive maintenance, Quality control, Real-Time Production Optimization in a Smart Factory 4.0.

Another interesting issue in ICSs for future study is the change of the normal behaviour over time due to, for example, IoT devices are aging. Hence new behaviour of devices can be classified as an anomaly, which then causes a high false-positive rate. It is the common issue for any anomaly detection approach using one-class classification Machine Learning in which we use only normal data for training. In fact, there is a research direction in online machine learning that updates old models with the most up-to-date data. And to the best of our knowledge, this research direction seems not have been investigated in FL. In the future, we would like to study deeper in the direction of online Federated Learning.

Besides, another critical aspect for our future work is: when manufacturing

sites are not identical, for example, in the number of machines. It raises the issue of imbalanced distributed training for the VAE-LSTM models running on the edge. Since data sets of different sites can be different, it increases the bias of pattern learned by the FL server. In the state of the art, there is very new research [49] that dealt with this issue to prevent the bias of training caused by imbalanced data distribution, so we presume we can learn the way to improve this problem from that work.

References

[1] Y. Lu, X. Xu, L. Wang, Smart manufacturing process and system automation – a critical review of the standards and envisioned scenarios, Journal of Manufacturing Systems 56 (2020) 312–325. doi:https://doi.org/10.1016/j.jmsy.2020.06.010.
URL https://www.sciencedirect.com/science/article/pii/S027861252030100X

[2] J. Wang, Y. Ma, L. Zhang, R. X. Gao, D. Wu, Deep learning for smart manufacturing: Methods and applications, Journal of Manufacturing Systems 48 (2018) 144–156, special Issue on Smart Manufacturing. doi:https://doi.org/10.1016/j.jmsy.2018.01.003.
URL https://www.sciencedirect.com/science/article/pii/S0278612518300037

[3] L. Wang, From intelligence science to intelligent manufacturing, Engineering 5 (4) (2019) 615–618. doi:https://doi.org/10.1016/j.eng.2019.04.011.
URL https://www.sciencedirect.com/science/article/pii/S2095809919301821

[4] N. Tuptuk, S. Hailes, Security of smart manufacturing systems, Journal of Manufacturing Systems 47 (2018) 93–106. doi:https://doi.org/10.1016/j.jmsy.2018.04.007.

URL https://www.sciencedirect.com/science/article/pii/S0278612518300463

[5] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, X. Yang, A survey on the edge computing for the internet of things, IEEE Access 6 (2018) 6900–6919. doi:10.1109/ACCESS.2017.2778504.

[6] D. P. Kingma, M. Welling, Auto-encoding variational bayesarXiv:1312.6114.

[7] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (8) (1997) 1735–1780. arXiv:https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf, doi:10.1162/neco.1997.9.8.1735.
URL https://doi.org/10.1162/neco.1997.9.8.1735

[8] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, S. Roberts, Anomaly detection for time series using vae-lstm hybrid model, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 4322–4326. doi:10.1109/ICASSP40776.2020.9053558.

[9] D. Myers, S. Suriadi, K. Radke, E. Foo, Anomaly detection for industrial control systems using process mining, Computers & Security 78 (2018) 103–125. doi:https://doi.org/10.1016/j.cose.2018.06.002.
URL https://www.sciencedirect.com/science/article/pii/S0167404818306795

[10] P. Nader, P. Honeine, P. Beauseroy, l_p-norms in one-class classification for intrusion detection in scada systems, IEEE Transactions on Industrial Informatics 10 (4) (2014) 2308–2317. doi:10.1109/TII.2014.2330796.

[11] A. Gumaei, M. M. Hassan, S. Huda, M. R. Hassan, D. Camacho, J. Del Ser, G. Fortino, A robust cyberattack detection approach using optimal features of scada power systems in smart grids, Applied Soft Computing

96 (2020) 106658. doi:https://doi.org/10.1016/j.asoc.2020.106658.
URL https://www.sciencedirect.com/science/article/pii/S1568494620305962

[12] P. Priyanga S, K. Krithivasan, P. S, S. Sriram V S, Detection of cyber-attacks in industrial control systems using enhanced principal component analysis and hypergraph-based convolution neural network (epca-hg-cnn), IEEE Transactions on Industry Applications 56 (4) (2020) 4394–4404. doi:10.1109/TIA.2020.2977872.

[13] G. Li, Y. Shen, P. Zhao, X. Lu, J. Liu, Y. Liu, S. C. Hoi, Detecting cyberattacks in industrial control systems using online learning algorithms, Neurocomputing 364 (2019) 338–348. doi:https://doi.org/10.1016/j.neucom.2019.07.031.
URL https://www.sciencedirect.com/science/article/pii/S0925231219309762

[14] D. Li, K. Paynabar, N. Gebraeel, A degradation-based detection framework against covert cyberattacks on scada systems, IISE Transactions 53 (7) (2021) 812–829. arXiv:https://doi.org/10.1080/24725854.2020.1802537, doi:10.1080/24725854.2020.1802537.
URL https://doi.org/10.1080/24725854.2020.1802537

[15] M. Kravchik, A. Shabtai, Efficient cyber attack detection in industrial control systems using lightweight neural networks and pca, IEEE Transactions on Dependable and Secure Computing (2021) 1–1doi:10.1109/TDSC.2021.3050101.

[16] F. Adamsky, M. Aubigny, F. Battisti, M. Carli, F. Cimorelli, T. Cruz, A. Di Giorgio, C. Foglietta, A. Galli, A. Giuseppi, F. Liberati, A. Neri, S. Panzieri, F. Pascucci, J. Proenca, P. Pucci, L. Rosa, R. Soua, Integrated protection of industrial control systems from cyber-attacks: the atena approach, International Journal of Critical Infrastructure Protection 21 (2018) 72–82. doi:https://doi.org/10.1016/j.ijcip.2018.04.004.

URL https://www.sciencedirect.com/science/article/pii/S1874548217301798

[17] R. F. Babiceanu, R. Seker, Cyber resilience protection for industrial internet of things: A software-defined networking approach, Computers in Industry 104 (2019) 47–58. doi:https://doi.org/10.1016/j.compind.2018.10.004.
URL https://www.sciencedirect.com/science/article/pii/S0166361517306954

[18] A. Essien, C. Giannetti, A deep learning model for smart manufacturing using convolutional lstm neural network autoencoders, IEEE Transactions on Industrial Informatics 16 (9) (2020) 6069–6078. doi:10.1109/TII.2020.2967556.

[19] A. Luca, M. G. C. Cimino, G. Manco, E. Ritacco, G. Vaglini, Using an autoencoder in the design of an anomaly detector for smart manufacturing, Pattern Recognition Letters 136. doi:10.1016/j.patrec.2020.06.008.

[20] R.-J. Hsieh, J. Chou, C.-H. Ho, Unsupervised online anomaly detection on multivariate sensing time series data for smart manufacturing, in: 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA), 2019, pp. 90–97. doi:10.1109/SOCA.2019.00021.

[21] J. Gao, X. Song, Q. Wen, P. Wang, L. Sun, H. Xu, Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks (2020). arXiv:2002.09545.

[22] R.-Q. Chen, G.-H. Shi, W.-L. Zhao, C.-H. Liang, A joint model for it operation series prediction and anomaly detection (2021). arXiv:1910.03818.

[23] L. Gjorgiev, S. Gievska, Time series anomaly detection with variational autoencoder using mahalanobis distance, in: V. Dimitrova, I. Dimitro-

vski (Eds.), ICT Innovations 2020. Machine Learning and Applications, Springer International Publishing, Cham, 2020, pp. 42–55.

[24] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, M. S. Hossain, Deep anomaly detection for time-series data in industrial iot: A ommunication-efficient on-device federated learning approach, IEEE Internet of Things Journal 8 (8) (2021) 6348–6358. doi:10.1109/JIOT.2020.3011726.

[25] T. T. Huong, T. P. Bac, D. M. Long, B. D. Thang, T. D. Luong, N. T. Binh, An efficient low complexity edge-cloud framework for security in iot networks, in: 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE), 2021, pp. 533–539. doi:10.1109/ICCE48956.2021.9352046.

[26] H.-D. Nguyen, K. P. TRAN, S. Thomassey, M. Hamad, Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management, International Journal of Information Managementdoi:10.1016/j.ijinfomgt.2020.102282.

[27] H. McMahan, E. Moore, D. Ramage, B. Agüera y Arcas, Federated learning of deep networks using model averaging.

[28] X. Wang, S. Garg, H. Lin, J. Hu, G. Kaddoum, M. J. Piran, M. S. Hossain, Towards accurate anomaly detection in industrial internet-of-things using hierarchical federated learning, IEEE Internet of Things Journal (2021) 1–1doi:10.1109/JIOT.2021.3074382.

[29] Y. Qin, H. Matsutani, M. Kondo, A selective model aggregation approach in federated learning for online anomaly detection, in: 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), 2020, pp. 684–691. doi:10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00119.

[30] Y. Ye, S. Li, F. Liu, Y. Tang, W. Hu, Edgefed: Optimized federated learning based on edge computing, IEEE Access 8 (2020) 209191–209198. doi:10.1109/ACCESS.2020.3038287.

[31] A. A. Cook, G. Mısırlı, Z. Fan, Anomaly detection for iot time-series data: A survey, IEEE Internet of Things Journal 7 (7) (2020) 6481–6494. doi:10.1109/JIOT.2019.2958185.

[32] M. F. Abdelaty, R. Doriguzzi Corin, D. Siracusa, Daics: A deep learning solution for anomaly detection in industrial control systems, IEEE Transactions on Emerging Topics in Computing (2021) 1–1doi:10.1109/TETC.2021.3073017.

[33] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber, Lstm: A search space odyssey, IEEE Transactions on Neural Networks and Learning Systems 28 (10) (2017) 2222–2232. doi:10.1109/TNNLS.2016.2582924.

[34] L. Gjorgiev, S. Gievska, Time series anomaly detection with variational autoencoder using mahalanobis distance, in: V. Dimitrova, I. Dimitrovski (Eds.), ICT Innovations 2020. Machine Learning and Applications, Springer International Publishing, Cham, 2020, pp. 42–55.

[35] P. Malhotra, L. Vig, G. M. Shroff, P. Agarwal, Long short term memory networks for anomaly detection in time series, in: ESANN, 2015.

[36] S. J. Sheather, J. S. Marron, Kernel quantile estimators, Journal of the American Statistical Association 85 (410) (1990) 410–416. doi:10.1080/01621459.1990.10476214.

[37] I. P. Turnipseed, A new scada dataset for intrusion detection research, 2015.

[38] E. Keogh, J. Lin, A. Fu, Hot sax: efficiently finding the most unusual time series subsequence, in: Fifth IEEE International Conference on Data Mining (ICDM'05), 2005, pp. 8 pp.–. doi:10.1109/ICDM.2005.79.
URL http://www.cs.ucr.edu/~eamonn/discords/

[39] Available, Nyc taxi and limousine commission, available, in: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, Last accessed on May, 2021.

[40] X. Liu, Chapter 14 - methods for handling missing data, in: X. Liu (Ed.), Methods and Applications of Longitudinal Data Analysis, Academic Press, Oxford, 2016, pp. 441–473. doi:https://doi.org/10.1016/B978-0-12-801342-7.00014-9.
URL https://www.sciencedirect.com/science/article/pii/B9780128013427000149

[41] R. Lopez Perez, F. Adamsky, R. Soua, T. Engel, Machine learning for reliable network attack detection in scada systems, in: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, pp. 633–638. doi:10.1109/TrustCom/BigDataSE.2018.00094.

[42] J. Shao, B. Zhong, Last observation carry-forward and last observation analysis, Statistics in Medicine 22 (15) (2003) 2429–2441. doi:https://doi.org/10.1002/sim.1519.

[43] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, H. Qiao, Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 187–196. doi:10.1145/3178876.3185996.
URL https://doi.org/10.1145/3178876.3185996

[44] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, S. Avestimehr, Fedml: A research library and benchmark for federated machine learning, arXiv preprint arXiv:2007.13518.

[45] A. in, https://docs.emqx.io/en/broker/v4.3/, Last accessed on May, 2021.

[46] A. in, https://github.com/tgraf/bmon, Last accessed on May, 2021.

[47] A. in, https://github.com/xybu/python-resmon, Last accessed on May, 2021.

[48] A. in, https://www.mediafire.com/folder/q2b8h079hpywq/UM25, Last accessed on May, 2021.

[49] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, L. Liang, Self-balancing federated learning with global imbalanced data in mobile systems, IEEE Transactions on Parallel and Distributed Systems 32 (1) (2021) 59–71. doi:10.1109/TPDS.2020.3009406.