Computers in Industry 140 (2022) 103692

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Light-weight federated learning-based anomaly detection for time-series data in industrial control systems

Huong Thu Truong[a,*], Bac Phuong Ta[b], Quang Anh Le[a], Dan Minh Nguyen[a], Cong Thanh Le[a], Hoang Xuan Nguyen[a], Ha Thu Do[c], Hung Tai Nguyen[a,*], Kim Phuc Tran[d]

[a] Hanoi University of Science and Technology, Viet Nam
[b] School of Electronic Engineering, Soongsil University, Seoul, Republic of Korea
[c] International Research Institute for Artificial Intelligence and Data Science, Dong A University, Danang, Viet Nam
[d] Univ. Lille, ENSAIT, ULR 2461 - GEMTEX - Génie et Matériaux Textiles, F-59000 Lille, France

ARTICLE INFO

Article history:
Received 22 January 2022
Received in revised form 16 April 2022
Accepted 19 April 2022
Available online 30 April 2022

Keywords:
Anomaly detection
ICS
Federated learning
Autoencoder
Transformer
Fourier

ABSTRACT

With the emergence of the Industrial Internet of Things (IIoT), potential threats to smart manufacturing systems are increasingly becoming challenging, causing severe damage to production operations and vital industrial assets, even sensitive information. Hence, detecting irregularities for time-series data in industrial control systems that should operate continually is critical, ensuring security and minimizing maintenance costs. In this study, with the hybrid design of Federated learning, Autoencoder, Transformer, and Fourier mixing sublayer, we propose a robust distributed anomaly detection architecture that works more accurately than several most recent anomaly detection solutions within the ICS contexts, whilst being fast learning in minute time scale. This distributed architecture is also proven to achieve lightweight, consume little CPU and memory usage, have low communication costs in terms of bandwidth consumption, which makes it feasible to be deployed on top of edge devices with limited computing capacity.

1. Introduction

Industrial control systems (ICS) (Bhamare et al., 2020) refers to many types of control systems and related instruments, which comprise devices, systems, networks, and controls used to operate and automate industrial processes. ICS devices and protocols are now deployed in every industrial sector and vital infrastructure, including manufacturing, transportation, energy, gas pipelines, water treatment, and so on. Besides, with the goal of improving the effectiveness of industrial production processes, the Industry is on the move towards Industrial Internet of Things (IIoT) infrastructure that connects multiple physical smart devices together. With the increasing use of operational technology (Corallo et al., 2022) in the context of IIoT-based Industry 4.0, an ICS can become the prime target for attacks from a variety of threats, for it contains valuable information affecting the performance of the whole industry. Many organizations such as the European Cyber Security Organization and the European Network and Information Security Agency have developed cybersecurity standards and gathered best practices to address this issue for the industrial control systems (Corallo et al., 2020). Therefore, it is crucial to maintain a close eye on these systems' behavior for attack events by using anomaly detection-based techniques.

Moreover, threats are becoming more complex, thus an anomaly detection solution that can promptly and correctly identify attacks whilst being light-weight enough to be implemented on devices with low computing capabilities in IoT-based industrial control systems is required.

Various anomaly detection methods for time-series data in the ICS context have been proposed, such as RNN (Sherstinsky, 2020), LSTM (Sherstinsky, 2020), and GRU (C. Xu et al., 2018). However, the performance of these sequence models still requires improvement. The primary problem with RNNs is that gradients are propagated over multiple stages, which tends to cause them to vanish or explode. For that reason, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have been widely adopted in time-series anomaly detection and have found particular success. Nevertheless, these algorithms still suffer from their sequential nature, as with RNN and its variants. Being inherently sequential hinders parallelization within samples, which slows down the training and inference processes, especially with long sequences.

* Corresponding authors.
E-mail addresses: huong.truongthu@hust.edu.vn (H.T. Truong), hung.nguyentai@hust.edu.vn (H.T. Nguyen).

https://doi.org/10.1016/j.compind.2022.103692
0166-3615/

To tackle those limitations, we propose a new hybrid solution of Autoencoder, Transformer and Fourier transform operating in a chain to enhance the detection performance, whilst reducing the training time and being more lightweight for more feasible deployment over edge devices of an IIoT-based industrial control system. This work can be considered as our effort to find a better practice for an efficient distributed cybersecurity architecture required by the resource constraints and business effectiveness of an IIoT-based ICS of a smart factory 4.0.

Unlike the other existing sequence-to-sequence models, Transformer does not employ Recurrent Networks. Instead, the Transformer model deploys an attention mechanism to extract dependency between each part of the incoming data simultaneously, thus making it very parallelizable. However, the attention mechanism usually requires high computation cost and large memory storage, so it might not be suitable for the distributed ICS contexts where distributed computing is handled at edge devices with limited hardware capacity. Hence, in our solution, we also make use of Autoencoder (AE) to reduce the dimension of input data whilst still preserving the most vital information. In addition, the Transformer model's running time is further sped up by replacing the attention layer in the Transformer block with a Discrete Fourier Transform (DFT) sublayer. By transforming the data from the time domain into the frequency domain, this Fourier transformation can capture the relationship between the input sequences similar to the attention mechanism (Lee-Thorp et al., 2021). DFT is a direct transformation that will result in a highly efficient and rapid approach since no learnable parameters are required. Besides, thanks to the implementation of the Fast Fourier Transform (FFT) algorithm, DFT reaches a substantially lower computational complexity of $\mathcal{O}(n \log n)$, compared to $\mathcal{O}(n^2)$ of the DFT calculation using the conventional matrix method.

From another perspective, ICS is a distributed system containing multiple components located on different machines that communicate and coordinate actions to synchronize as a single coherent system. Therefore, in this paper, we design an overall Federated learning (FL)-based anomaly detection architecture so called FATRAF. FL is one of the most promising and adaptive candidates for communication costs in a distributed environment. FATRAF is able to provide knowledge of other data patterns from other edge zones to each local model through a federated global model update. Furthermore, implementing anomaly detection tasks locally at each edge and federating those local models with FL improve the system response time upon attack arrivals since the detection model is conducted directly right at the edge, which is close to attack sources. The synchronization of the FL technique and an ML-based detection strategy aids in achieving both benefits of detection effectiveness and light-weight computing.

Overall, FATRAF brings the following advantages:

1. A light-weight local learning model for anomaly detection in ICS based on a new hybrid of Autoencoder - Transformer - Fourier to yield fast learning time and consume hardware resources reasonably. That makes FATRAF be feasibly implemented in practical distributed edge devices in an IoT-based ICS architecture.
2. An unsupervised learning model based on normal data only, along with a dynamical-threshold-determining scheme named Kernel Quantile Estimation (KQE) (Sheather and Marron, 1990) to tune the detection mechanism dynamically. Therefore, it is able to dynamically keep track of new anomalous patterns that may change over time in ICS, while yielding high detection performance for time-series data in industrial control systems, compared to state of the art solutions.
3. A federated learning (FL) framework to enable efficient distributed anomaly detection near anomaly/attack sources. Hence the system response time upon attacks can be improved. Distributed or edge computing helps blocking an infected zone without affecting the common operation of the entire system, therefore enhancing production efficiency. FL allows the edge sites to share model information with each other, aiming to optimize anomaly detection performance globally. In practice, this solves the lack of training data in each edge site, especially with multivariate and high-dimensional data sets.

In conclusion, FATRAF is a distributed anomaly detection architecture that can be deployed on top of the distributed IIoT-based Industrial Control Systems. The architecture supports to detect anomalies in the system quickly, thereby enhancing the system security of a smart factory. That makes the management of an ICS more robust and secure, ensuring its production line not be interrupted from attacks.

The rest of our paper is represented as follows: Section 2 describes state of the art in the field of anomaly detection for ICSs. Section 3 elaborates our design and integration of the whole solution - FATRAF - that yields an efficient detection performance whilst being light-weight, achieving faster training time and consuming fewer hardware resources. The performance evaluation and experiments are described in Section 4 in which we investigate the detection performance of FATRAF as well as its edge computing efficiency. Finally, conclusions are presented in Section 5.

2. Related work

Anomaly detection has always been a critical issue in Smart Manufacturing (SM), which requires timely detection and accuracy. In this section, we will describe the state of the art in this field. Among the existing solutions, the highlights of our design are that we propose an unsupervised learning method that can always keep track of changing in data patterns inside an ICS in a quite small timescale. The detection method is deployed in the Federated-learning manner to unite knowledge of multiple local detection learning models from distributed areas into a single global learning model. In each local learning model, we design Autoencoder, Transformer and Fourier transform in a sequence to create a very effective unsupervised learning detection which can classify time-series data accurately while reducing dramatically training time and computing cost which are the serious constraint for distributed security deployment over distributed IIoT-based infrastructure.

Solutions have been proposed to detect cyber attacks for ICS, such as Al-Abassi et al. (2020), Mokhtari et al. (2021), Kravchik and Shabtai (2021), Chang et al., and Anton et al. (2019). In Al-Abassi et al. (2020), the authors performed an ensemble method using a deep neural network and decision tree for attack identification. The hybrid model can also address commonly-encountered issues with imbalanced massive data sets. The performance is evaluated over ICS data sets, and the model provides the accuracy of 99.67% and 96.00% on the SWaT (itrust) and gas pipeline data sets (Turnipseed, 2022), respectively. However, the authors used all normal and attack data for the training process, resulting in higher accuracy but lack of adaptability when new attack patterns are employed. In the same direction of using supervised learning, work Mokhtari et al. (2021) proposed a measurement intrusion detection system approach to detect any common abnormal activity in an ICS system with the HAI data set (Hil-based augmented ics security dataset). They applied the well-known supervised learning algorithms such as K-Nearest Neighbor, Decision Tree, and Random Forest, with the last one having the best performance of 99.67%. However, the data in the ICS system is commonly in time series, and anomalous patterns change continuously. Although the performance is proven quite high, the model in Mokhtari et al. (2021) is incapable of extracting characteristics of time-series data; thus, the actual performance may differ. Furthermore, as we reproduce their proposed method with

the corresponding data set, we notice that the published classification performance appears to have been calculated with a micro-averaging strategy, which is inappropriate in binary classification problems, especially when the correct detection of instances of minority class (i.e., anomaly class) is crucial to the overall operation. Therefore, contrary to the opinion provided by work Mokhtari et al. (2021), we presume that using unsupervised training models with normal data will be a more effective method in detecting attacks and adapting to new abnormal behaviors. In addition, an anomaly detection framework with the combination of k-mean and convolutional neural network was proposed in work (Chang et al.). These techniques, however, do not provide optimal performance for time-series data. It can reach an accuracy of 95.53% and an F1-score of 89.08% with the gas pipeline data set, as the paper results show. In another aspect, work (Kravchik and Shabtai, 2021) combined some popular machine learning methods such as 1D-CNN, Undercomplete Autoencoder, and Principal Component Analysis with short-time Fourier transformation, transforming time-domain signals into frequency representation to remove noise and handle slow attacks. Similar to work (Chang et al.), the performance of these models has not been optimized with F1 scores of only 82–88% over the SwaT data set.

Since ICS data is frequently in time series, it is reasonable to use sequence models such as Long Short-term Memory (LSTM) and Gated Recurrent Unit (GRU) to capture temporal relationships in data sequences. Nevertheless, these solutions still suffer from their sequential nature. Being inherently sequential hinders parallelization within samples, which slows down training and inference processes, especially with long sequences. Therefore, work (Zarzycki and Ławryńczuk, 2021; Mothukuri et al., 2021; Perales Gómez et al., 2020; Li et al., 2019), for example, did not yield significant anomaly detection performance when employing LSTM and GRU for ICS time series data. In work (Zarzycki and Ławryńczuk, 2021), the author presented a data-driven predictive modeling approach for ICS systems. The models such as Recurrent Neural Network (RNN), LSTM, and GRU were used to detect anomalies. The best-achieved accuracy is 81.38% over the SCADA data set. To detect anomalies with decentralized on-device data, work (Mothukuri et al., 2021) presented a Federated Learning (FL)-based anomaly detection strategy for IoT networks based on the combination of GRUs and LSTM. However, the proposed method's performance is insufficient; the accuracy in each FL client reaches the highest performance of 95.5%. Work (Perales Gómez et al., 2020) provided a methodology called MADICS for Anomaly Detection in Industrial Control Systems using a semi-supervised anomaly detection paradigm. The performance of MADICS in terms of Recall is slightly low over its testing data sets. In work (Li et al., 2019), the author proposed MAD-GAN to deal with the lack of labeled data using an supervised method - Generative Adversarial Networks (GANs) and LSTM-RNN. This model considered correlation between the spatial and temporal characteristics of multivariate ICS data. However, the experimental results received over the SWaT and WADI data sets indicate no trade-off between the measures such as Precision and Recall, while the F1-score remains low in general. Furthermore, the authors of Li et al. (2019) did not account for the model training time, which is the major drawback of LSTM and GRU. Work (Huong et al., 2021) proposed a cyberattacks detection mechanism using the combination of Variational Autoencoder (VAE) and LSTM. Although the detection performance is considerably high, the model in Huong et al. (2021) faces difficulty in terms of running time since the LSTM block requires a long training time. Liu et al. (2021) presented an attention CNN-LSTM model within a FL framework for anomaly detection in IIoT edge devices. However, since ICS data usually contains multiple features (the Gas Pipeline and SWaT data sets have 17 and 51 features respectively), this complex design may necessitate more computation and training time.

In order to tackle those shortcomings of the LSTM and GRU networks, Google launched a novel, fully promising architecture known as Transformer (Vaswani et al., 2017) in 2017, an encoder-decoder architecture based on an attention mechanism rather than RNN. Although initially evolved in the field of Natural Language Processing, this architecture has been deployed in various anomaly detection applications. As an example, the spacecraft anomaly detection research in Meng et al. (2022) demonstrated that their transformer-encoder-based framework, with the adoption of the attention mechanism and the masking strategy, could conserve time cost up to roughly 80% in comparison with LSTM, while only reaching an F1 score of 0.78 on the NASA telemetry data set. From our standpoint, this figure needs to be further ameliorated since the capture of all actual abnormalities should be prioritized, which aims to minimize the damage in the worst case. Similarly, by introducing an Anomaly-Attention mechanism so as to measure the association discrepancy between anomalous and normal samples, an unsupervised time series anomaly detection proposal called Anomaly Transformer in Xu et al. (2021) shows considerable prediction capability over the service monitoring data set - Pooled Server Metrics (PSM). Nonetheless, regarding the application in industrial water treatment, that study achieves a modest F1 score of 94.07% and Recall of 96.73% on the SWaT data set, so further improvement is needed.

With respect to anomaly detection problems in IoT contexts, we also find that the transformer-inspired solution in a recent work (Kozik et al., 2021) brings remarkable performance. By leveraging a transformer encoder followed by a two-layer feed-forward neural network, the classifier deployed in Kozik et al. (2021) performs well on the Aposemat IoT-23 data set (Garcia et al., 2020) with the best F1 score of 95%. Nevertheless, as aforementioned, the performance of such a classifier becomes degraded when a new abnormal pattern or behavior arises since it is trained with both defined normal and abnormal observations (i.e, labeled samples). Meanwhile, our proposed solution is trained with completely normal data, attempting to find non-conform patterns in the data during inference, which results in effective anomaly detection performance, even with unknown abnormalities or attacks. Work (Kozik et al., 2021), furthermore, lacks a comprehensive assessment in terms of runtime as well as feasibility on actual hardware devices if deployed in distributed ICS systems, which prompts us to conduct evaluation scenarios for our solution itself.

Moreover, to the best of our knowledge on the relevant studies, our FATRAF approach is considered at the forefront of anomaly detection deployment for IIoT-based ICS ecosystems that can cover many related issues. It is the first work that combines Federated learning, AE, Transformer and Fourier transform to deal with the cybersecurity issue, especially for IIoT-based ICSs. The approach provides high detection performance within the context of IoT distributed computing, more lightweight to be more feasibly deployed over limited hardware capacity of edge devices in an IIoT-based network, and faster training time to meet the requirement on real-time updating to cope with changes in attack patterns from various sources.

3. System architecture design

Before we describe our Federated Learning-based solution design in which the local data will have the same feature vectors, let us introduce its applicability in a real distributed industrial control system. We can take an example of a distributed environment in an IIoT-based factory with many workshops/zones or maybe many different factories under the same company that wants anomaly detection applications to perform predictive maintenance. Centralizing all this data on a data warehouse or data lake increases the risk that attackers are able to tap into information about

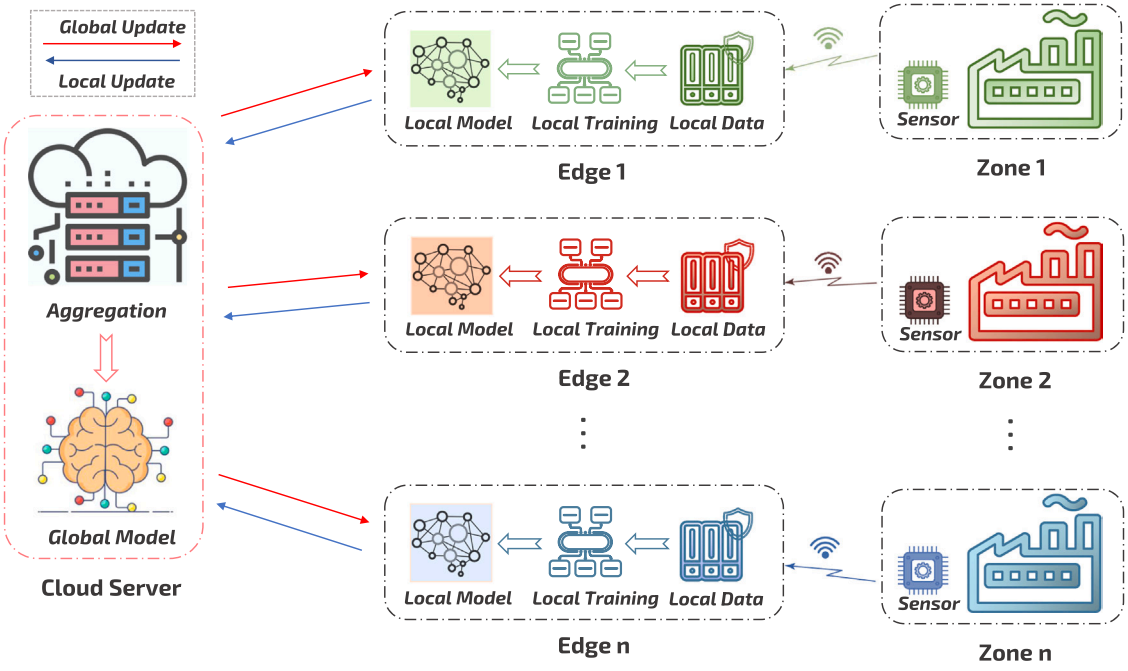


Fig. 1. FATRAF architecture.

manufacturing processes or even problems at individual companies. In this case, we can perform predictive maintenance by placing sensors to collect vibration signals from electric motors (of the same type), from which to build an early anomaly detection model to perform predictive maintenance before failures occur. In other words, we can predict the remaining useful life (RUL) for the electric motors and switch them into maintenance a few cycles before we think a failure will happen. It will help the company to reduce maintenance costs by not performing maintenance too early (that causes production interruption); or by not performing maintenance late which leads to failures. In addition, our method can be applied so that electric motor manufacturers can provide predictive maintenance services to their customers without having direct access to the motors' operating data as the operating companies consider this data confidential.

3.1. System architecture overview

In this paper, we propose an Anomaly Detection (AD) architecture for time-series data in Industrial Control Systems (ICSs) which provides a fast and efficient learning model combined with Federated learning for distributed computing. The overall proposed architecture is named **FATRAF** (**F**ederated Learning-based **A**utoencoder-**Tra**nsformer-**F**ourier Anomaly Detection). Our goal is to design an AD system that has a fast training time and is light weight to accommodate frequent learning update, while still either retaining the same or improving the detection performance in comparison with some existing AD solutions for ICSs in the literature.

As illustrated in Fig. 1, FATRAF comprises two main components: .

- **Edge sites:** In an IIoT-based smart factory monitored by industrial control systems, there are various manufacturing zones, in which sensor systems are installed to gather readings that signify operating states over time. Subsequently, the time-series data, as the local data, is transmitted wirelessly to edge devices in the vicinity of the corresponding manufacturing zones. Designated to monitor anomalies, these edge devices employ the local data as inputs for training its own local anomaly detection model - ATRAF (**AE**-**Tra**nsformer-**F**ourier learning model). This deployment allows detecting anomalies timely right at the edge sites, making use of the computing capacity of edge devices, and distributing heavy computation tasks that could overload the cloud server.
- **Cloud Server:** The cloud server undertakes two primary functions: system initialization and aggregation of local models sent from different edges. The whole process of model aggregation and global model updating down to all local models are called Federated Learning.

In the following sections, we will describe how the local anomaly detection module is designed and implemented to accelerate the learning speed, be more light-weight to adapt well to the limited computing capacity of edge hardware.

3.2. Design of the ATRAF detection model at the edge

In the FATRAF architecture, the anomaly detection model deployed at each edge site is designed as the hybrid learning model of Autoencoder (AE), Transformer, and Fourier - called **ATRAF**. The mission of this design is to create a light-weight, low-computing, and fast-learning model that can yield high detection performance for time-series data, while still ensuring fast training time to cope with the requirement of frequently re-updating the learning model. This requirement comes from the fact that devices of a factory can be aging, or attack/anomaly behaviors could change over time. Therefore, in order to catch up with any new data pattern on time, the security system has to keep learning and updating the learning model quite frequently. That leads to another requirement that any learning model deployed on those distributed edge devices with limited computing capacity should be light-weight to work fast and consume less computing resources.

Overall, ATRAF is an unsupervised learning model that relies totally on normal data so as to attempt to detect abnormal patterns during inference, thereby enhancing anomaly detection performance. This suits the fact that novel abnormalities are unknown, or the data is not always labeled, thereby solving the limitation of some

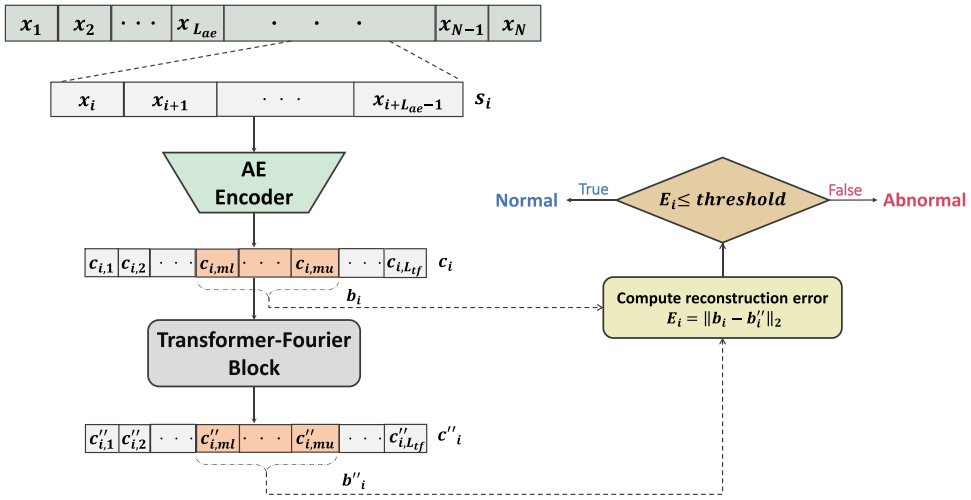


Fig. 2. ATRAF learning model operation during the testing process.

classifiers such as the recent transformer-based work (Kozik et al., 2021) for IoT applications.

Autoencoder has the advantage of capturing the distinct structural and temporal regularities of the data in each distributed zone. The ability to capture data temporal regularities makes this model suitable for anomaly detection for time series data. In addition, AE is a good method to reduce the data dimensionality that results in a shorter learning time. As depicted in Fig. 2, the Autoencoder block is first trained locally, then only the encoder part is kept for later phases in our design.

As the output of the AE block's Encoder, compressed time-series data is then fed into the Transformer-Fourier block to estimate the long-term correlation of the data sequences and extract a distinguishable and meaningful criterion, which is used to determine irregularities in the data. Transformer (Vaswani et al., 2017) is used in our design as a remedy for other time-series approaches such as LSTM and GRU since the Transformer model processes sequences in parallel. This method achieves high anomaly detection performance for time-series data and better running time.

In addition, by replacing a Self-Attention sublayer of a pure Transformer with a Fourier mixing sublayer, the training process of the Transformer - Fourier block is significantly accelerated in comparison with, for example, that of the recurrent neural networks such as LSTMs, while retaining or even improving detection performance. This is due to the fact that the Transformer-Fourier model processes sequences in parallel, instead of sequentially.

In short, the Transformer-Fourier block comprises two main layers, namely one Transformer encoder layer and only one Fourier layer, which is depicted concretely in the Section 3.2.2. The Fourier layer has a lighter memory footprint since it replaces the self-attention sublayer in the Transformer encoder with an unparameterized Fourier transform sublayer. Furthermore, the Transformer-Fourier block proves to be more efficient than the Transformer encoder with the same number of layers, since it maintains the same level of performance as the Transformer encoder while shortening the training process, making it well-suited for resource-constrained edge devices. This statement is demonstrated through our experiments in Section 4.

In the following sections, we will describe in detail the implementation of the ATRAF model.

3.2.1. Design of the autoencoder block

Autoencoder (AE) is a symmetrical, unsupervised neural network with a "bottleneck" in the central hidden layer, which has fewer nodes than the input and output layers. It is trained to reconstruct the output as closely as possible to the input. After this process, the network has learned to compress the data into a lower-dimensional code and rebuild the input from it. Generally, an AE consists of 3 components: .

- **Encoder**: An encoder is a feed-forward, fully connected neural network that compresses the input into a lower-dimensional code.
- **Code**: Code, also known as the latent-space representation, is a compact "summary" or "compression" of the input. This code keeps the most essential information of the input while employing fewer features.
- **Decoder**: Decoder is also a feed-forward network and has a similar structure to the encoder. This network is in charge of reconstructing the input back to the original dimensions from the code.

In our design, the AE block is first trained locally at the edge devices for a number of epochs. After that, the decoder part is discarded while the encoder part is kept for compressing the input into a lower-dimensional code. As illustrated in Fig. 3, we construct the encoder and decoder with only two fully-connected hidden layers each. The goal of this approach is to accomplish the simplicity and light-weight of the model, allowing it to be trained on edge devices with limited computing capacity, as well as to reduce communication costs while still providing sufficient detection performance. AEs, in particular, learn a map from the input to themselves via a pair of encoding and decoding stages.

$$X' = Decoder(Encoder(X))$$

Where: X and X' are the input and output of AE.

- As depicted in Fig. 2, the input $X = \{x_1, x_2, \ldots, x_N\}$, where $x_n \in \mathbb{R}^{d_{hidden}}$, $n = \overline{1 \div N}$, is divided into $\kappa = N - L_{ae} + 1$ overlapping sequences $\{s_1, s_2, \ldots s_\kappa\}$, where $s_i = \{x_i, \ldots, x_{i+L_{ae}-1}\}$, $i = \overline{1 \& \#x00f7; \kappa}$, is a sequence of length L_{ae}.
- These sequences are then used to train the local AE model as depicted in Fig. 3. Denote the output of the Encoder corresponding to the input s_i as $c_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,L_{tf}}\}$ where $L_{tf} \leq L_{ae}$. Denote the output of the Decoder corresponding to d_i as $s'_i = \{x'_i, \ldots, x'_{i+L_{ae}-1}\}$. We train the local AE model by using the loss function of the $\mathscr{L}_2$ norm $\|s'_i - s_i\|_2$.
- After the training process, the sequences are fed through the AE model's encoder, producing compressed code sequences $\{c_1, c_2, \ldots, c_\kappa\}$ to train the Transformer-Fourier block, where $c_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,L_{tf}}\}$.

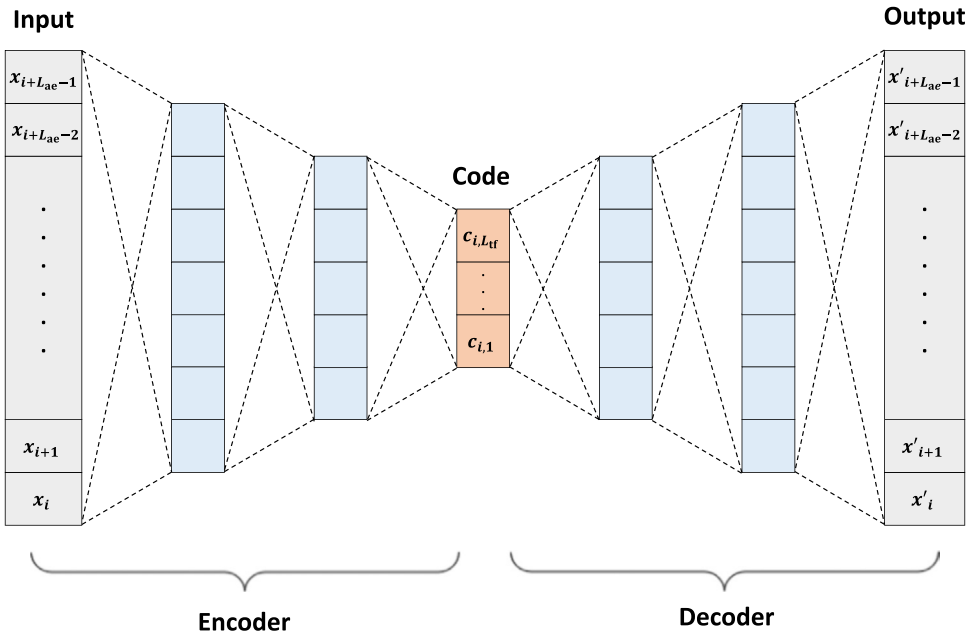


Fig. 3. Autoencoder configuration.

In order to reduce the computation cost as well as improve the detection performance, the length of the input sequence L_{ae}, the code sequence L_{tf} of the AE model are chosen through a pragmatic process of different configurations of L_{ae} and L_{tf} in order to achieve a good F1-score.

By fixing parameter L_{ae} at a time while varying L_{tf}, we found that the detection performance (i.e., F1 score) as well as the training time increases as L_{ae} increases and decreases with the ratio $\frac{L_{ae}}{L_{tf}}$. The final set of parameters L_{ae} and L_{tf} is chosen based on a grid search on a range of values for L_{ae} and L_{tf} in order to find the best F1-score in the trade off with running time. By limiting the maximum value of L_{ae} to 200, the model can achieve a good performance while maintaining acceptable training and inference time on our edge devices.

3.2.2. Design of the transformer-Fourier block

In order to overcome the problem of sequential computing and take advantage of GPU parallel computing to speed up the learning process, the attention mechanism was adopted to capture long-term dependencies.

In this paper, a design of the Transformer with the mixing Fourier Transform sublayer is proposed as a learning model that relies on an attention mechanism to draw temporal dependencies in sequences. To speed up the Transformer encoder architecture, the unparameterized simple linear Fourier Transformation is integrated to replace the self-attention sublayer of the Transformer encoder. This linear mixing sublayer was proven to work efficiently in terms of learning speed while maintaining a comparable performance, especially at long input lengths (Lee-Thorp et al., 2021).

The hybrid model of Transformer and Fourier transform mixing sublayer can also provide a smaller memory footprint and faster runtime than an all-attention-sublayer Transformer on GPU. As a result, it is much more suitable for resource-constrained edge devices.

Fig. 4 depicts comprehensively the Transformer-Fourier block, which is shown in Fig. 2. Its operation will be described in detail below.

3.2.2.1. Positional encoding. Each output code sequence c_i of the local encoder - AE model, is passed through the Positional Encoding layer, which injects relative positional information into the sequences. Since the sequences are processed in parallel, we need to ensure that the model is able to differentiate between the positions in the code sequences. In this paper, we add the sine and cosine encodings of variable frequencies to each sequence $c_i = \{c_{i,1}, c_{i,2}, ..., c_{i,L_{tf}}\}$.

$$PE_{p,2m} = \sin\left(\frac{p}{10^{\frac{8m}{d_{hidden}}}}\right) \tag{1}$$

$$PE_{p,2m+1} = \cos\left(\frac{p}{10^{\frac{8m}{d_{hidden}}}}\right) \tag{2}$$

Where: .

- $p \in \{1, 2, ..., L_{tf}\}$ is the position
- $m \in \{0, 1, ..., \lfloor d_{hidden}/2 \rfloor\}$ is the dimension index
- d_{hidden} is the size of the code sequences' hidden dimension.

3.2.2.2. Transformer layer. The Transformer encoder layer consists of a Self-Attention sublayer, followed by a Position-wise Feed Forward sublayer. The Transformer encoder can learn the inter-position dependencies of the non-blocked portions of the input sequences, producing an embedding $c'_i = \{c'_{i,1}, c'_{i,2}, ...c'_{i,L_{tf}}\}$ of the same dimensionality as the input sequence c_i.

3.2.2.2.1. Masked self-attention sublayer. The Masked self-attention sublayer structure is depicted in Fig. 5. This sublayer takes the sum of Positional Encoding and the code sequence c_i as the input:

$$input = c_i + PE \tag{3}$$

and calculates the Query (Q), Key (K) and Value (V) matrices as follows:

$$Q = input \times w_q + \beta_q \tag{4}$$

$$K = input \times w_k + \beta_k \tag{5}$$

$$V = input \times w_v + \beta_v \tag{6}$$

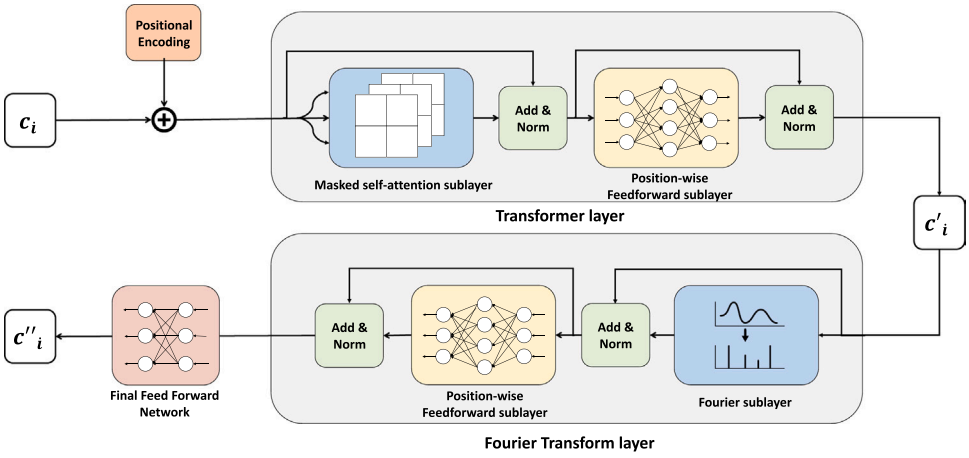


Fig. 4. The transformer-Fourier block.

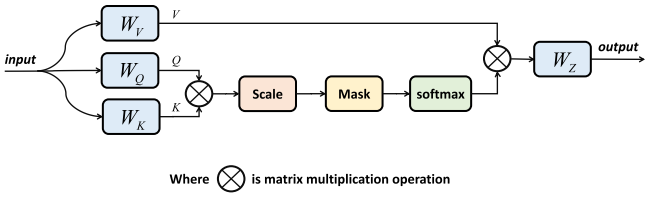


Fig. 5. The masked self-attention sublayer.

where (w_q, β_q), (w_k, β_k), (w_v, β_v) are weight and bias matrices of W_Q, W_K, W_V nodes in Fig. 5 respectively.After that, the scaled dot-attention sublayer computes the dot product of two matrices Q and K and then divides the result with d_{hidden} to obtain *score* of value V.

$$score = \frac{Q \times K^T}{\sqrt{d_{hidden}}} \tag{7}$$

Next, we apply a mask to *score* in order to block a contiguous sub-sequence of c_i, from index *ml* to *mu* (as illustrated in Fig. 2). This mask is achieved by assigning the respective score in the self-attention matrix to minus infinity. Finally, *softmax function* is applied to *score* to get the final normalized weight of value V before being passed into node W_Z to produce *output* which is attention information of sequence c_i. These last operations can be summarized as follows:

$$output = softmax(mask(score)) \times V \times w_z + \beta_z \tag{8}$$

where (w_z, β_z) are weight and bias matrices of W_Z node.

As indicated in Fig. 4, there is always an *Add&Norm* operation after each sublayer in Transformer-Fourier Block. This operation performs normalization on the sum of input and output of the preceding sublayer:

$$Add\,\&\,Norm() = LayerNorm(input + output) \tag{9}$$

3.2.2.2. Position-wise feed forward sublayer. The output of the Self-Attention sublayer and the following Add&Norm (for now let's say τ_i) is fed directly through a Position-wise Feed Forward network, which applies the following transformation to each position of the input identically and independently.

$$PWFF(\tau_i) = ReLU(\tau_i w_1 + \beta_1)w_2 + \beta_2 \tag{10}$$

Where (w_1, β_1) and (w_2, β_2) are parameters of two linear transformations. The input and output layers of the Position-wise Feed Forward network have the same number of neural nodes of d_{hidden}. Again, another *Add&Norm* operation is applied as in Eq. (9), resulting in sequence c'_i as follows:

$$c'_i = LayerNorm(\tau_i + PWFF(\tau_i)) \tag{11}$$

3.2.2.3. Fourier transform layer. After that, the embedding $c'_i \in \mathbb{R}^{L_{tf} \times d_{hidden}}$ is passed into the Fourier Transform layer, in which a non-parameterized Fourier sublayer is added to replace the traditional Self-Attention sublayer of the Transformer encoder. Upon this embedding representation, two one-dimensional Discrete Fourier Transforms (DFT), as a mixing method instead of the Self-Attention mechanism, are applied along the sequence dimension (i.e, L_{tf}) and the hidden dimension (i.e, d_{hidden}) of c'_i. The obtained results are then kept the real parts. Theoretically, the DFT transforms a sequence of Θ discrete numbers $\{\mathbf{a}_\theta\} \coloneqq a_0, a_1, \ldots, a_{\Theta-1}$ to another sequence of discrete numbers, $\{\mathbf{A}_\eta\} \coloneqq A_0, A_1, \ldots, A_{\Theta-1}$, which is defined by:

$$DFT\{a\} = \mathbf{A}_\eta = \sum_{\theta=0}^{\Theta-1} a_\theta e^{-\frac{ED2\pi}{\Theta}\eta\theta} \tag{12}$$

Accordingly, the output of the Fourier sublayer is expressed as follows:

$$Output_{fourier-sublayer} = \Re(DFT_{hidden}\{DFT_{sequence}\{c'_i\}\}) \tag{13}$$

In our design, the DFT is computed with the Fast Fourier Transform (FFT) algorithm, which accelerates the computation time to $\mathcal{O}(n \log n)$. Moreover, thanks to its parameter-free characteristic, the Fourier transform is regarded as a relatively effective and lightweight mixing method compared to the attention mechanism.

The remaining sublayer operations within the Fourier Transform layer, namely Position-wise Feedforward and *Add&Norm* as shown in Fig. 4, are identical to those of the Transformer layer which are discussed above.

3.2.2.4. Final feed forward network. This is the last layer of the architecture, which applies a ReLU-activated linear transformation to the output of the Fourier layer to produce a reconstructed sequence of c_i which is denoted $c''_i = \{c''_{i,1}, c''_{i,2}, \ldots c''_{i,L_{tf}}\}$. The input and output of this feed forward network also have d_{hidden} neural nodes.

3.2.3. Error and threshold computation

The reconstruction errors are calculated between the masked sub-sequence of c_i, i.e., $b_i = \{c_{i,ml}, \ldots, c_{i,mu}\}$ and the corresponding reconstructed sub-sequence $b''_i = \{c''_{i,ml}, \ldots, c''_{i,mu}\}$ as demonstrated in Fig. 2.

$$E_i = \|b_i - b''_i\|_2 \tag{14}$$

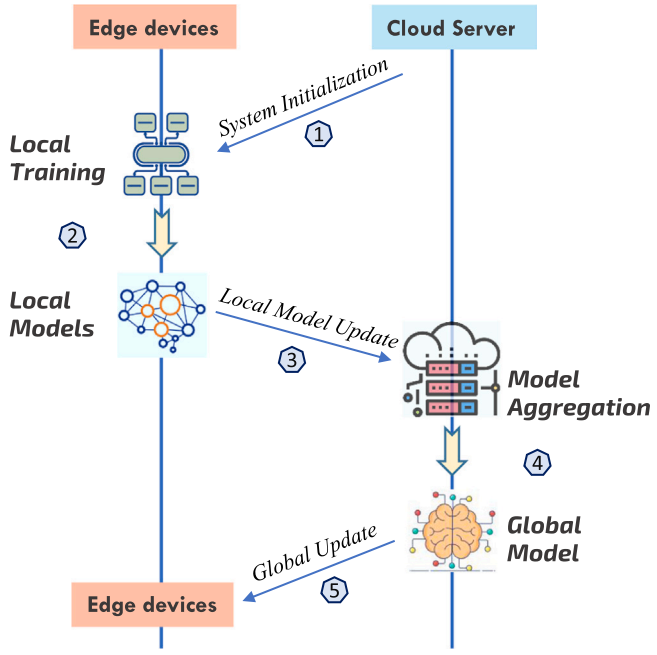


Fig. 6. Flow diagram of FATRAF.

An anomaly threshold λ_{Th} is determined by the Kernel Quantile Estimation (KQE) technique (Sheather and Marron, 1990) that is used on the reconstruction errors to classify the input sequence $s_i = \{x_i, ..., x_{i+L_{ae}-1}\}$. For convenience of annotation, we sort the set of reconstruction errors achieved in Eq. (14) in a non-decreasing order $E_{(1)} \leq E_{(2)} \leq ... \leq E_{(\iota)} \leq ... \leq E_{(\kappa)}$. The anomaly threshold is then determined by:

$$\lambda_{Th} = \sum_{\iota=1}^{\kappa} \left[\int_{\frac{\iota-1}{\kappa}}^{\frac{\iota}{\kappa}} \frac{1}{\Omega} \mathscr{K}\left(\frac{t-\rho}{\Omega}\right) dt \right] E_{(\iota)} \tag{15}$$

where $\mathscr{K}$ is the density function, chosen as the standard Gaussian kernel $\mathscr{K}(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})$; the asymptotically optimized bandwidth $\Omega = \sqrt{\frac{\rho(1-\rho)}{\kappa+1}}$ controls the estimator's smoothness, and ρ $(0 < \rho < 1)$ is the preset value.

3.3. FATRAF operation in the proposed architecture

This section describes the Federated Learning process of the FATRAF architecture. As Fig. 6 depicts, the procedure can be summarized into the following key stages:

1. *System Initialization:* At the beginning, the cloud server establishes a global model with specific learning parameters and sending it to each edge device of corresponding edge site.
2. *Local Training:* After receiving the initial configuration, by utilizing the on-site data collected from sensors, edge devices conduct a local training process for their Autoencoder blocks. Subsequently, at each edge site, the local training data is fed into the encoder part of the trained Autoencoder model so as to obtain corresponding embeddings or code sequences. These code sequences are then applied to train the Transformer-Fourier block.
3. *Local Model Update:* After the local training, each edge device sends the learnable parameters of the Transformer-Fourier block w^r_{t+1} to the cloud server for aggregation. Specifically, w^r_{t+1} includes weight and bias matrices from W_Q, W_K, W_V, W_Z nodes in the Masked Self-attention sublayer and all linear transformations in the Position-wise Feedforward sublayer along with Final Feedforward Network.
4. *Model Aggregation:* After deriving trained weights w^r_{t+1} from all edge clients, the cloud server federates them and constructs a new global model version by the formula proposed by McMahan et al. (2016), as follows:

$$w_{t+1} = \sum_{r=1}^{n} \frac{v_r}{v} w^r_{t+1} \tag{16}$$

where: n: the number of edge sites
v_r: amount of data of the rth edge site
v: total amount of data of all edge sites
w^r_{t+1}: local learnable parameters of the Transformer-Fourier block at the rth edge site at time $t+1$
w_{t+1}: global learnable parameters at time $t+1$.
Although there are several proposals on the federated learning technique to aggregate local models to a single global model such as Karimireddy et al. (2020), Li et al. (2020), through our various experiments, the approach of federating local weights presented in Eq. (16) is still found to be the most efficient in terms of detection performance.
5. *Global Update:* Finally, the cloud server broadcasts back the new configuration w_{t+1} to each edge device as to update the local models of Transformer-Fourier block. In the subsequent learning rounds, edge devices use the updated models to continue training and the communication process will repeat in order to optimize the local Transformer-Fourier models until the global learning model converges.

4. Experiments and evaluation

In this section, we focus on evaluating the performance of FATRAF in terms of detection performance in different scenarios as well as the implementation feasibility in the Edge Computing environment.

First, we briefly introduce different time-series data sets in our experiments and the data preprocessing. Second, we also describe our testbed settings and tools. Finally, anomaly detection performance and edge computing efficiency will be studied throughout our experiments.

4.1. Data sets and pre-processing

In our experiments, we compare the performance with some mostly up-to-date reference anomaly detection solutions for ICS that run on top of some other time-series or non-purely time-series data sets. We take the data sets presented in Table 1 as the main ICS use cases. In addition, to cross validate our solution detection performance in diverse contexts, we utilize other time-series data sets of disparate areas as listed in Table 2.

In fact, all data sets need to be pre-processed before being fed into the ATRAF learning model. As for the SCADA Gas Pipeline data set, it initially carries many missing values (i.e., "?" values) throughout the entire data set. To deal with this problem, we make use of the Last Observation Carried Forward (LOCF) method (Shao and Zhong, 2003), which uses the immediately previous value within the same field to substitute the missing values. In the case of the SWaT and HAI data sets, as their raw data sets contain many correlated features, using all of them in our proposed learning model will increase the computational burden considerably, resulting in exceptionally long training time. For that reason, we perform feature selection to reduce the number of features presented in each data point by investigating the correlation between them: features with high correlation are removed; features with zero variance are discarded.

To provide more information, we summarize the statistical details about the pre-processed data sets in Table 3. It should be noted

Table 1
List of the data sets used for main use cases.

Data sets	Description
Gas Pipeline (Turnipseed, 2022)	Time-series data set was collected in 2015 from the Supervisory Control and Data Acquisition (SCADA) gas pipeline system by Mississippi State University Lab. Each data point has 17 features, containing network information as well as a payload which gives details about the gas pipeline's state and settings.
SWaT (itrust)	Secure Water Treatment, launched in 2015, is the data set collected from a scaled down water treatment plant. Being applied in secure Cyber Physical Systems research, SWaT collects the plant's continuous operation data in 11 days, in which 7 days of normal activities and 4 days under attacks. Each sample the SWaT data set contains 51 features from different sensors and actuators.
HAI (Hil-based augmented ics security dataset)	HIL-based Augmented ICS data set stemmed from a practical ICS testbed augmented with a Hardware-In-the-Loop (HIL) simulator, introduced for ICS anomaly detection research. The testbed aims to emulate steam-turbine power and pumped-storage hydropower generation. Initially published in 2020, the time-series data set includes 59 features recorded under normal and abnormal (in case of attacks or system failures) behaviors.
Power Demand (Keogh et al., 2022)	Univariate time-series data set, including 1-year-long power consumption readings of a Dutch research facility in 1997.

Table 2
List of the data sets used for cross validation.

Data sets	Description
ECGs (Keogh et al., 2022)	Time-series of the heartbeat electrical signals
Respiration (Keogh et al., 2022)	Measurements of patient's respiration when waking up by thorax extension
Gesture (Keogh et al., 2022)	2-feature data set indicates the right hand's coordinates while performing different actions
Space shuttle (Keogh et al., 2022)	Solenoid current's measurements of a Marotta MPV-41 series valve cycled on and off
NYC taxi (Nyc taxi and limousine commission)	Information on New York taxi passenger data stream (Jul 2014–Jun 2015)

that after being pre-processed, all training samples are of normal class, i.e, we have removed all abnormal sequences in the original training sets. For this reason, only the percentages of the abnormal samples in the test sets are shown. When used for training Federated Learning- solutions, training sets are further split into 4 smaller data subsets, corresponding to 4 distributed local zones. These 4 subsets are divided successively to preserve the time-series characteristics, forming 4 consecutive time-series sequences containing only samples of the normal class. In addition, we also use validation sets when training the models, which are divided from the training sets with a split ratio of 1:9. For evaluation process, we assess the performance of trained models using test sets containing both normal and anomalous data sequences, described in Table 3.

features to be in the same scale between 0 and 1 according to the min-max normalization described in Eq. (17):

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{17}$$

where x_i, x'_i are before and after values of the feature. x_{max}, x_{min} are the maximum and minimum values of that feature.

4.2. Testbed setup and implementation

In our experiments, the ATRAF learning model is implemented in two learning modes: Centralized Learning and Federated Learning. As the whole FATRAF architecture is designed to be light-weight, parallel computing of the ATRAF detection model should be leveraged. Therefore, the training and inference tasks are implemented in GPU-equipped devices. In the Centralized-learning mode, the experiment is conducted using:

- a Dell Precision 3640 Tower workstation featuring an NVIDIA Quadro P2200 GPU and an Intel Core i7-10700 K CPU with 16 GB RAM.

In Federated-learning mode, the experiment is conducted using:

- 4 NVIDIA Jetson Nano B01 boards to emulate 4 edge devices (standing for 4 different distributed zones). Local learning models are trained with local data subsets at each edge.
- A ThinkSystem SR550 (Intel Xeon Silver 4210, 64 GB RAM) to serve as the Cloud Server for aggregation tasks (i.e., Federated learning).
- WiFi interfaces for the edges and the Cloud to communicate their weight matrices via the MQTT protocol. MQTT is proven to be a light-weight, reliable and scalable protocol for IoT networks. For this purpose, the open-source EMQ X Broker is hosted in the Aggregation Server.

For the performance measurement purposes, some tools are used as follows:

- Tool *bmon* (bmon) is used to measure bandwidth occupation in each link between the Edge- link in the Federated Learning- architecture.
- Built-in utility *tegrastats* keeps track of computational resources usage as well as energy consumption on edge devices during

Table 3
Statistics of datasets after pre-processing.

Dataset	Train	Test	Dimensions	Anomalies (%)
Power Demand	18,145	14,786	1	15.24
Gas Pipeline	160,869	68,658	16	21.77
HAI	379,885	444,600	14	3.94
SWaT	495,000	449,919	16	12.14
Space shuttle - TEK14	2009	1992	1	30.02
Space shuttle - TEK16	3067	951	1	10.41
Space shuttle - TEK17	2119	926	1	4.75
Respiration - nprs43	6213	7411	1	12.20
Respiration - nprs44	12,592	11,127	1	8.07
Gesture	6180	3000	2	24.63
Nyc taxi	5500	4820	1	0.10
ECG - Chfdbchf01275	1833	1841	2	14.61
ECG - chfdbchf1345590	2439	1287	2	12.35
ECG - chfdbf15	10,863	3348	2	4.45
ECG - ltstdb2022143	2610	1121	2	11.50
ECG - ltstdb20321240	2011	1447	2	9.60
ECG - mitdb100180	2943	2255	2	8.38
ECG - qtdbsel102	34,735	9882	2	2.01
ECG - stdb3080	2373	2721	2	9.52
ECG - xmitdbx1080	3152	1756	2	21.58

training and testing tasks. This program extracts real-time information about the usage of CPU, RAM, GPU, and the energy consumption of the NVIDIA Jetson Nano board.

The implementation of the ATRAF model is written in Python 3.8.10 using the PyTorch framework (release 1.9.0). We also leverage FedML (He et al., 2020) framework to realize the Federated Learning setting.

4.3. Performance evaluation

4.3.1. Detection performance evaluation

To assess the detection performance of FATRAF, some common standard metrics are used such as Precision, Recall and F1-Score. The definition of these metrics are defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where:

- *TP*: True Positive represents samples which are correctly classified as positive class.
- *TN*: True Negative represents samples which are correctly classified as negative class.
- *FP*: False Positive represents samples which are incorrectly classified as positive class.
- *FN*: False Negative represents samples which are incorrectly classified as negative class.

To calculate the performance metrics in our experiments, we apply the simple point-adjust approach proposed in H. Xu et al. (2018), which assumes it is adequate to trigger a malfunction or attack alert within any subset of an anomaly window. As we detect anomalies in windows of time-series data points, a full window is viewed as anomalies in the ground truth label if it contains at least one anomalous point. When the model detects any part of this window as anomalous, the whole window is also considered as correctly detected as anomalies.

It is also noted that during the evaluation process, we prioritize Recall and overall F1-Score metric over Precision. This priority can be explained that in real-life scenarios, detecting all actual attacks and malfunctions in the system is often more critical. The cost of the system operator in case of being attacked is too high; thus, a small number of false alarms is tolerable

4.3.1.1. Experiment 1 – federated learning vs. centralized learning. Federated Learning technique enables efficient distributed computing, wherein edge devices are responsible for detecting malfunctions or attacks in their own local zones, leading to faster response time. By letting each zone leveraging others' model information, this learning mode also enhances production efficiency with other benefits such as taking advantage of edge capabilities, alleviating computing workload on central server, reducing occupied bandwidth for data exchange as well as better isolating anomalous zones. However, each edge site tends to have much smaller set of data, this could induce inadequate anomaly detection models in comparison with the approach of centralized learning. Our first experiment aims to compare the detection performance of FATRAF with its centralized computing mode (i.e. implementing the ATRAF learning model in the centralized cloud mode) to determine if our Federated Learning- solution is fit to be implemented in the ICS environment.

Table 4 illustrates the performance of the two modes over the 4 time-series data sets of ICSs such as Power Demand, HAI, SWaT, and Gas Pipeline data sets. The detection performance of the centralized mode slightly outperforms the Federated Learning mode. This can be obviously explained that the learning-based model in the Federated Learning mode just learns from its own smaller local data set, so we will have to trade off a bit of detection performance with the benefits of Federated learning.

For cross-validation, the detection performance of both learning manners is verified with other time-series data sets which do not belong to the ICS context such as Space shuttle, Respiration, Gesture, NYC taxi, ECG. As Table 4 shows, the performance of FATRAF mostly approximates the Centralized-learning mode of ATRAF in all cross-validation data sets. However, ATRAF in both modes is proven to detect anomalies efficiently.

After testing, we notice that each client should train its model for only 1 local epoch within one round as to reduce overall training time. In our experiments, FATRAF training process is performed among clients for a number of communication rounds until convergence is achieved. As an illustration, Fig. 7 shows the reconstruction error on training set (training error) and validation set (validation error), computed according to Eq. (14), on the Gas Pipeline data set of one client participating in FATRAF after the local

Table 4
FATRAF vs. its centralized learning mode.

Data set	Centralized			FATRAF		
	Precision	Recall	F1	Precision	Recall	F1
Power Demand	0.9339	0.9797	0.9563	0.9285	0.9442	0.9363
Gas Pipeline	0.9699	1	0.9847	0.9683	1	0.9839
HAI	0.9039	0.9973	0.9483	0.8939	1	0.9440
SWaT	0.9404	0.9871	0.9632	0.9389	0.9775	0.9578
Space shuttle - TEK14	0.9874	1	0.9936	0.9738	1	0.9867
Space shuttle - TEK16	0.9851	1	0.9925	0.9296	1	0.9635
Space shuttle - TEK17	0.9728	1	0.9862	0.9728	1	0.9862
Respiration - nprs43	0.9567	0.9627	0.9597	0.9578	0.9730	0.9654
Respiration - nprs44	0.9174	0.9224	0.9199	0.9195	0.9799	0.9488
Gesture	0.9339	0.9921	0.9621	0.9331	0.9906	0.9610
Nyc taxi	0.8837	1	0.9382	0.9419	1	0.9701
ECG - Chfdb_chf01_275	0.9761	1	0.9879	0.9761	1	0.9879
ECG - chfdb_chf13_45590	0.9810	1	0.9904	0.9773	1	0.9885
ECG - chfdbf15	0.9118	1	0.9538	0.9118	1	0.9538
ECG - ltstdb_20221_43	0.9841	1	0.9920	0.9841	1	0.9920
ECG - ltstdb_20321_240	0.9558	1	0.9774	0.9714	1	0.9855
ECG - mitdb__100_180	0.9536	1	0.9763	0.9474	1	0.9730
ECG - qtdbsel102	0.7990	1	0.8883	0.7891	1	0.8821
ECG - stdb_308_0	0.9547	1	0.9768	0.9521	1	0.9755
ECG - xmitdb_x108_0	0.9795	1	0.9896	0.9856	1	0.9927

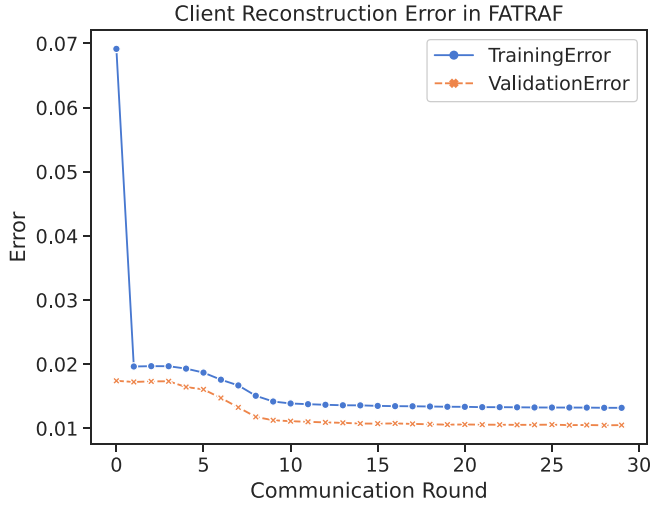


Fig. 7. Reconstruction loss during the FATRAF training based on data set Gas Pipeline for 30 communication rounds.

training phase of AE is finished. It can be seen that the model hardly reduces its validation error, the error decrease is less than 10^{-4} after communication round 10. When this criterion of 10^{-4} for validation error is satisfied for all clients, the FATRAF training process stops. For the remaining data sets used in our experiments, the number of communication rounds varies with the maximum being 50 rounds for some cross-validation data sets.

4.3.1.2. Experiment 2 – performance of federated Learning based approaches. In this experiment, the detection performance of FATRAF is compared with a recent detection work for ICS that applies Federated Learning called FL-VAE-LSTM (Huong et al., 2021). As Table 5 shows, FATRAF improves the detection performance in all metrics: Precision, Recall, F1-score with both of the ICS data sets: Power Demand and Gas Pipeline. It also outperforms FL-VAE-LSTM in most of the cross-validation time-series data sets. Unlike relatively fluctuating detection performance of FL-VAE-LSTM across some tested data sets, FATRAF demonstrates a little more consistent trend of high performance in anomaly detection capability. While with LSTM networks in the FL-VAE-LSTM solution, time steps influence others through their preceding and succeeding time steps due to its sequential nature, the relationship between a time step and its distant time steps may not be as straightforward as in FATRAF where direct relationship is achieved by the attention mechanism or DFT as the mixing method. This could affect some use cases where long sequences are utilized.

In conclusion, the results show that FATRAF can detect anomalies efficiently and stably in ICS systems that have time-series data. Later in the following subsection, the running time of FATRAF and FL-VAE-LSTM (Huong et al., 2021) is also compared to show that the training time of FATRAF much outperforms FL-VAE-LSTM.

4.3.1.3. Experiment 3 – comparison with other existing AD solutions for ICS over different contexts. The third experiment focuses on the comparison of ATRAF in both the Centralized and Federated Learning mode and several most recent works (Al-Abassi et al., 2020; Mokhtari et al., 2021; Xu et al., 2021) in the field of anomaly detection in ICS.

Fig. 8 depicts the detection performance of the 3 solutions: SAE proposed in Al-Abassi et al. (2020), our FATRAF and its centralized computing mode on 2 different data sets: Gas Pipeline (Turnipseed, 2022) and SWaT (itrust). As these data sets are primary case studies in Al-Abassi et al. (2020), using them in the detection performance comparison will yield better validation results. Whilst FATRAF gets marginally lower results than its centralized counterpart as shown in Experiment 1, it can be seen that on the Gas Pipeline data set, FATRAF outperforms SAE on all three metrics with Precision, Recall, F1-Score being 0.9683, 1, 0.9839, respectively, as opposed to 0.9463, 0.9372, 0.9383 achieved by SAE.

On the contrary, the experiment on the SWaT data set shows that FATRAF a bit underperforms SAE's roughly perfect performance in Precision, Recall, F1-Score of 0.97, 0.99, and 0.99, respectively. This can be explained as SWaT is not a purely time-series data set whilst FATRAF is designed to cope with time-series effectively.

Another comparison is made between our proposed solution and the measurement intrusion detection system (MIDS) introduced in Mokhtari et al. (2021), which utilizes 3 machine learning techniques such as KNN, Decision Tree, and Random Forest. As can clearly be seen from Table 6, the detection performance of both FATRAF and its centralized mode remarkably outperforms MIDS with all three different detection techniques. One point worth mentioning is that Precision, Recall, and F1-Score values are yielded 0 when using MIDS over the HAI data set. This outcome can be explained as the classification algorithms are not able to detect the minority class, which is the anomaly class in the test set. In fact, the original paper (Mokhtari et al., 2021) presented much higher classification performance, approximately ideal results, than our shown experimental results. The reason could be that those performance metrics may have been

Table 5
Federated-learning approaches over the time-series data sets.

Data set	FL-VAE-LSTM (Huong et al., 2021)			FATRAF		
	Precision	Recall	F1	Precision	Recall	F1
Power Demand	0.7355	0.9100	0.8135	0.9285	0.9442	0.9363
Gas Pipeline	0.9609	0.9982	0.9792	0.9683	1	0.9839
Space shuttle - TEK14	0.8623	0.8431	0.8536	0.9738	1	0.9867
Space shuttle - TEK16	1	1	1	0.9296	1	0.9635
Space shuttle - TEK17	0.9650	1	0.9822	0.9728	1	0.9862
Respiration - nprs43	0.9313	0.5530	0.6939	0.9578	0.9730	0.9654
Respiration - nprs44	0.5347	0.5027	0.5182	0.9195	0.9799	0.9488
Gesture	0.5278	1	0.6910	0.9331	0.9906	0.9610
Nyc taxi	0.9606	1	0.9799	0.9419	1	0.9701
ECG - Chfdb_chf01_275	0.9175	1	0.9570	0.9761	1	0.9879
ECG - chfdb_chf13_45590	0.9489	1	0.9738	0.9773	1	0.9885
ECG - chfdbf15	0.9458	1	0.9721	0.9118	1	0.9538
ECG - ltstdb_20221_43	1	1	1	0.9841	1	0.9920
ECG - ltstdb_20321_240	1	1	1	0.9714	1	0.9855
ECG - mitdb__100_180	1	1	1	0.9474	1	0.9730
ECG - qtdbsel102	0.9604	1	0.9797	0.7891	1	0.8821
ECG - stdb_308_0	0.6073	0.6373	0.6220	0.9521	1	0.9755
ECG - xmitdb_x108_0	1	0.7628	0.8654	0.9856	1	0.9927

(a) Gas Pipeline data set (b) SWaT data set

Fig. 8. Detection performance of SAE (Al-Abassi et al., 2020) vs. FATRAF vs. ATRAF's centralized mode on two datasets.

Table 6
FATRAF and ATRAF's centralized mode vs. measurement intrusion detection system (Mokhtari et al., 2021).

Data set		HAI			Gas pipeline		
		Precision	Recall	F1	Precision	Recall	F1
MIDS (Mokhtari et al., 2021)	KNN	0.7254	0.1685	0.2735	0.4236	0.6791	0.5218
	Decision Tree	0	0	0	0.5113	0.7076	0.5937
	Random Forest	0	0	0	0.4913	0.7587	0.5964
Centralized ATRAF		0.9039	0.9973	0.9483	0.9699	1	0.9847
FATRAF		0.9404	0.9871	0.9632	0.9683	1	0.9839

calculated when classifying an imbalanced test set with a micro-averaging strategy which highly encourages the classifier to focus on the dominant class for the trade-off of the minority class. However, this strategy is irrelevant in the context of binary classification tasks. Also, correctly detecting minority classes should be the priority when dealing with anomaly detection problems.

Finally, Fig. 9 shows the detection performance differences of proposed solutions and Anomaly Transformer (Xu et al., 2021) which also applies the Transformer model for time-series anomaly detection tasks. The comparison is drawn from their performance on the SWaT data set. It can be noted from the chart that our proposed detection mechanism is slightly better than Anomaly Transformer whose Precision, Recall and F1-Score metrics are 0.9155, 0.9673, 0.9407, respectively.

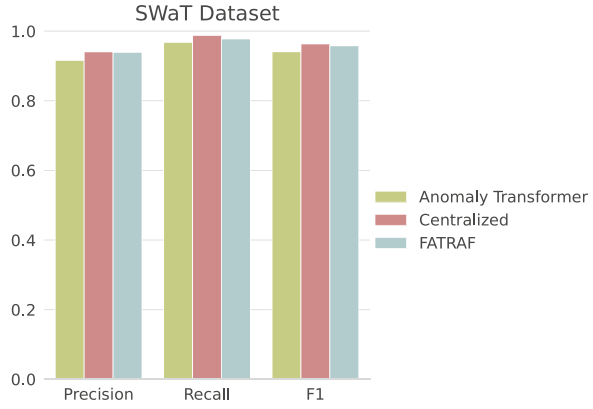


Fig. 9. FATRAF and ATRAF's centralized mode vs. anomaly transformer (Xu et al., 2021).

4.3.2. Edge computing evaluation

Since our proposal aims to be deployed on resource-constrained IoT edge devices, we should validate if FATRAF not only reaches a remarkable performance as demonstrated above but also saves edge resources effectively, partly thanks to its light-weight feature. Consequently, this section dives into evaluating the edge computing performance in terms of memory usage, GPU, CPU usage, power consumption, running time, and bandwidth occupation.

4.3.2.1. ***Memory usage***. In this measurement, we can see how the memory of each edge device is used during the training phase of the FATRAF on the Gas Pipeline data set. As described above, the training phase comprises two continuous phases: local training of Autoencoder and training of Transformer-Fourier in the federated environment. The edge computing assessment is carried out with 10 local epochs of training the Autoencoder and 10 communication rounds of training the Transformer-Fourier block. Fig. 10 demonstrates the memory usage of an NVIDIA Jetson Nano representing an edge device during the whole operation of 1155 s (roughly 19 min). As it can be noticed, the Autoencoder phase takes about 410 s or nearly 7 min, consuming around 70% of total memory. After that, the client continues to feed data through the trained encoder to create input for the Transformer-Fourier training phase. For the latter phase, the Jetson client increases its constant memory usage to approximately 75% of its total of 4 GB RAM.

It should be noted that the memory space is common between GPU and CPU, there is no dedicated GPU memory in NVIDIA Jetson Nano. Therefore, this memory usage is not for training model purposes only, it is also used by the operating system and other general-purpose programs.

4.3.2.2. CPU usage. In order to see how CPU is occupied for the training phase of the learning model, and to analyze how light-

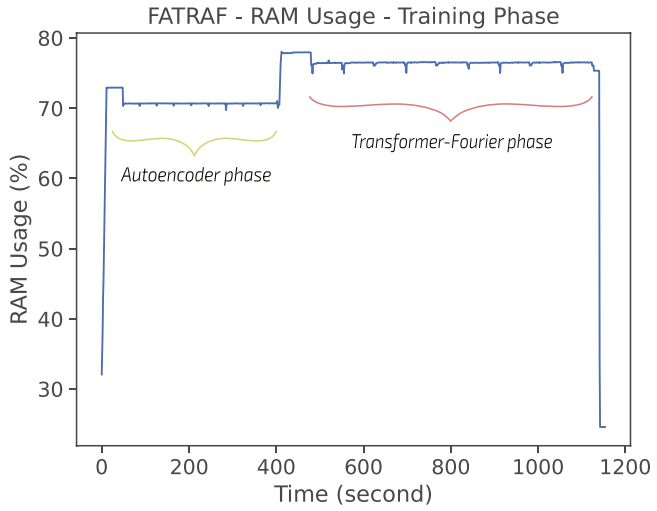


Fig. 10. Memory usage of an edge device (NVIDIA Jetson Nano) during the FATRAF training phase.

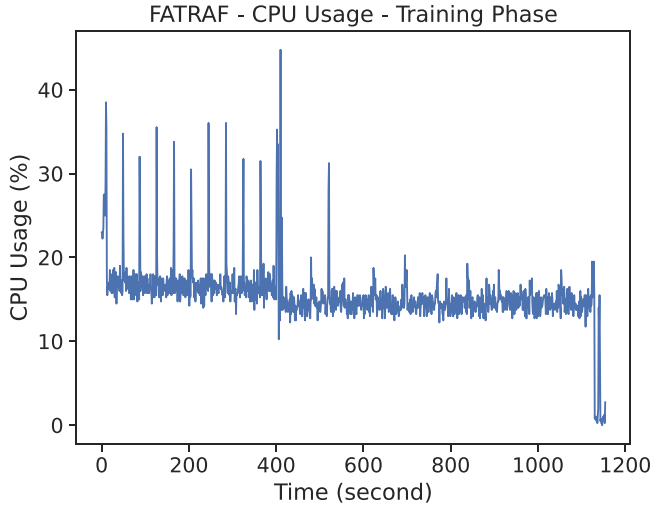


Fig. 11. CPU usage of an edge device (NVIDIA Jetson Nano) during the FATRAF training phase.

weight our proposed solution is, CPU usage is measured and presented in Fig. 11. As the proposed detection mechanism leverages parallel computing and GPU training, the learning model just makes a spike of around 40% of the CPU usage, but mostly consumes less than 20% of the hardware CPU during the training phase. This result shows that our proposed learning model is quite light-weight which is suitable for being deployed in an edge-computing environment.

4.3.2.3. GPU usage. The GPU usage of the edge hardware based on NVIDIA Jetson Nano is illustrated in Fig. 12. Since the GPU hardware is primarily responsible for the training detection model, the GPU usage of Jetson is utilized to its maximum during the FATRAF operation in order to accelerate the learning process of both the Autoencoder and Transformer-Fourier block.

In ICS environment, many other production tasks may be deployed at the edge devices, each requires different resources usage on the system (GPU, CPU, memory). To accommodate those tasks, the system resources occupied by FATRAF can be restricted to a certain lower level at the cost of correspondingly longer training time.

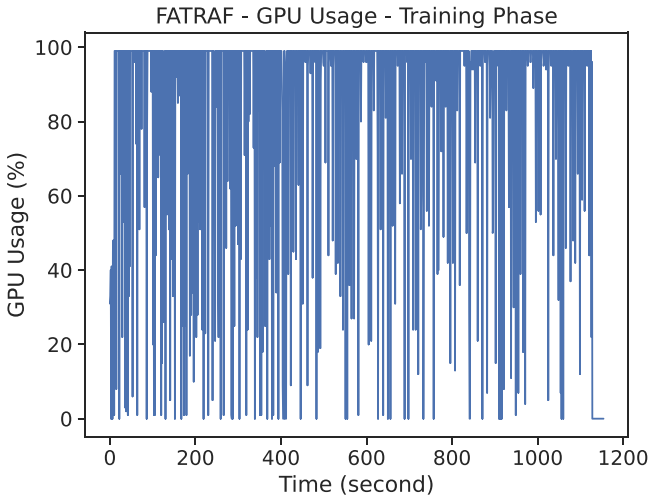


Fig. 12. GPU usage of NVIDIA Jetson nano during FATRAF training phase.

4.3.2.4. Running time evaluation. In this experiment, the training time of the FATRAF learning model will be compared with the FL-VAE-LSTM model proposed in the recent work (Huong et al., 2021) of the same field and context.

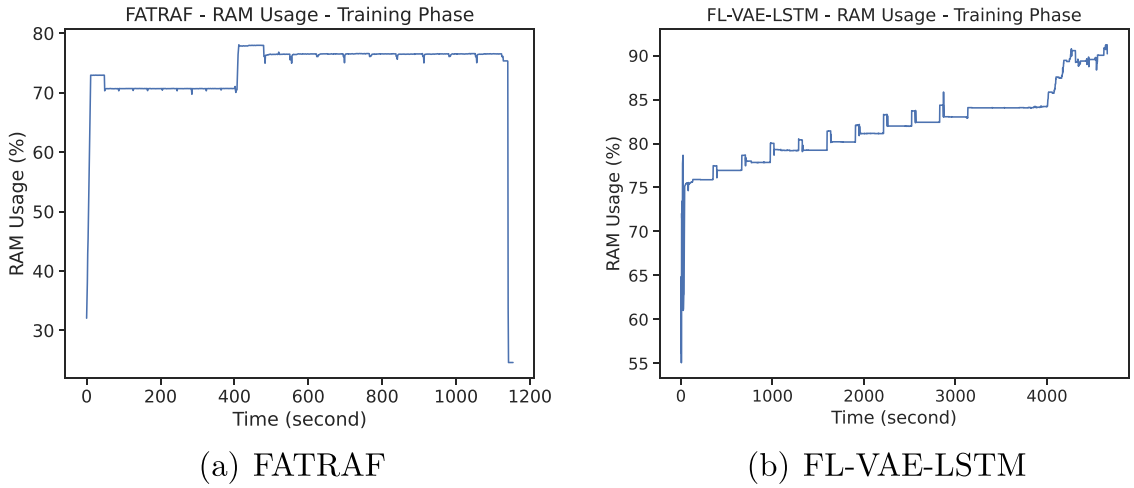
Whilst the settings for the FATRAF operation remains the same as described the above measurements, the training of the detection mechanism FL-VAE-LSTM is configured with 10 communication rounds in the VAE and LSTM phase each (as the convergence point), on top of the same Gas Pipeline data set.

It is also worth mentioning that both of the detection learning models are trained on the GPU device of NVIDIA Jetson Nano. As Fig. 13 illustrates, when training the 2 models on the same hardware, the training time of the hybrid of AE - Transformer - Fourier takes 1200 s overall, whilst the combination of VAE - LSTM needs around 5000 s to finish the training process. Such low training time (20 min) makes FATRAF more suitable for deployment in ICS without prolonged interruption of manufacture.

In conclusion, in this research, we have achieved our goal of reducing the training time of the learning model, which, as the results, fits much better to the IoT environment.

In addition, we extend our experiment to demonstrate the running time performance of the Hybrid Transformer-Fourier that replaces the attention sublayer with the Fourier transform sublayer. The Fourier transform is applied to reduce the training time as well as the complexity, making the mechanism more light-weight. As dealing with diverse problems, more Fourier layers may be employed to enhance the detection performance instead of traditional transformer encoder layers. To better understand the benefit of hybrid Transformer-Fourier over all-attention Transformer, Fig. 14 illustrates the relationship of how training time of these models is increased in proportion to the number of layers ranging from 2 to 10. The recorded time involves 10 rounds of training over the Gas Pipeline data set, after the Autoencoder phase. As the number of layers increases, learning time of the hybrid Transformer-Fourier model is proven to be more efficient than one of the all-attention transformer model.

4.3.2.5. Bandwidth consumption. Leveraging the Federated Learning technique, in FATRAF, each client sends all learnable parameters of the Transformer-Fourier block to the cloud server in each round. To be specific, these learnable parameters, including weight and bias matrices, are of W_Q, W_K, W_V, W_Z nodes from the Masked Self-attention sublayer and all linear transformations from the Position-wise Feedforward sublayer as well as Final Feedforward Network. Fig. 15 and Fig. 16 depict the bandwidth occupied in the upstream



(a) FATRAF (b) FL-VAE-LSTM

Fig. 13. Memory usage and training time of FATRAF vs. FL-VAE-LSTM (Huong et al., 2021).

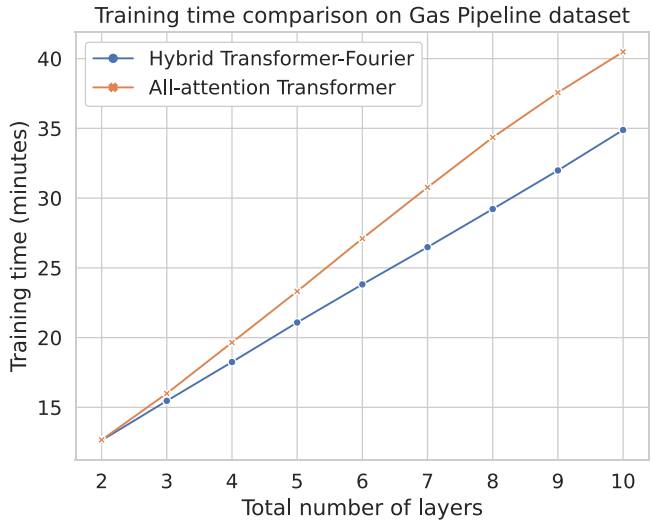


Fig. 14. Training time of hybrid transformer-Fourier model vs. all-attention-sublayer transformer model.

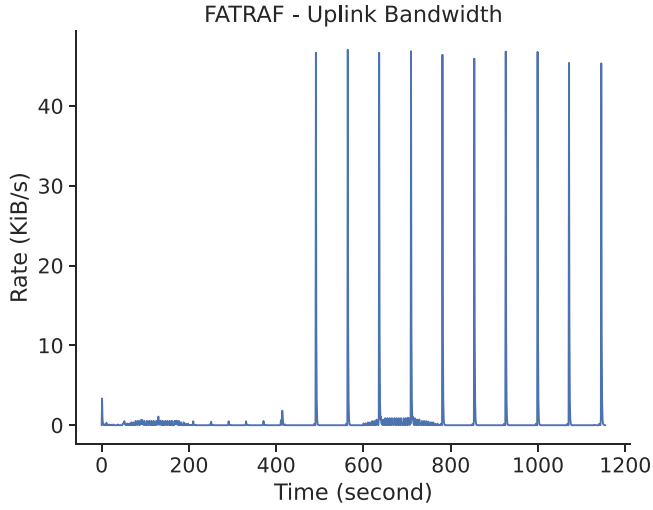


Fig. 15. Bandwidth occupation in the edge-cloud upstream link of FATRAF.

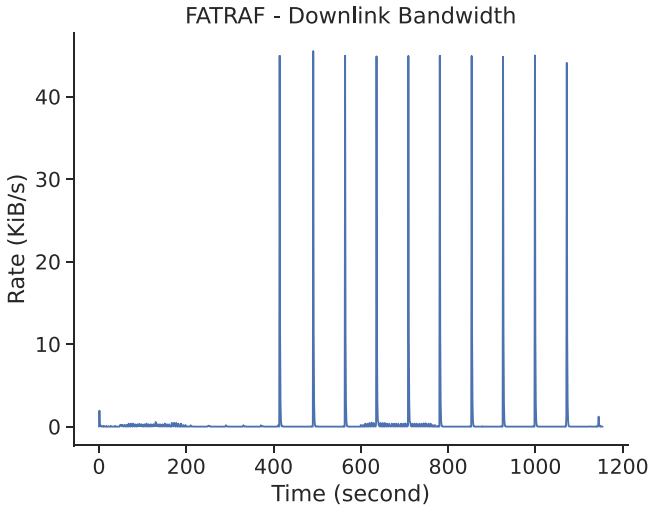


Fig. 16. Bandwidth occupation in the edge-cloud downstream link of FATRAF.

and downstream link by each client in the edge-cloud transmission. Because the Autoencoder training phase is entirely implemented within the local site, the clients do not exchange any information in this phase as illustrated in the first part of the graph. On the contrary, when the client enters the Transformer-Fourier training phase, in every communication round, the model's weight matrices are transferred to the cloud for federation (i.e., aggregation) when the local training is done. As can be seen in Fig. 15 and Fig. 16, there are 10 spikes of bandwidth occupation of around 45 kiB/s. Each spike spans no more than 2 s and corresponds to one transmission or reception of weight matrices in one round at the edge device. Such low communication cost will spare bandwidth for other information transmission in Industrial IoT systems, thus increasing the feasibility of the FATRAF architecture.

4.3.2.6. Power consumption. To have a deeper insight on the performance and influence of our proposed model from the various aspects, power consumption at each edge device during the training phase is taken into account. In fact, power consumed in each electronic component within Industrial systems could also be a critical factor too, taking into account energy efficiency and keeping the earth green. Fig. 17 shows a sketch of how FATRAF consumes power during each phase in the whole training duration. With an

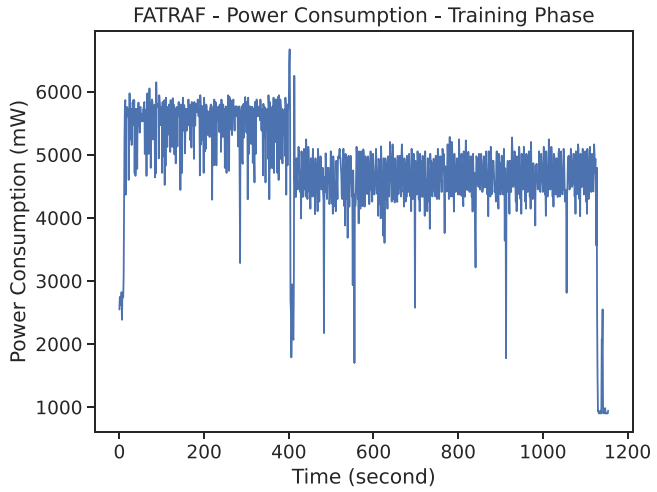


Fig. 17. Power consumption of NVIDIA Jetson nano during the FATRAF training phase.

Table 7
Training time and bandwidth occupation with different system parameters.

d_{hidden}	L_{ae}	L_{tf}	Training time	Uplink (kiB/s)		Downlink (kiB/s)	
				Avg	*Max*	*Avg*	*Max*
1	200	100	10 min18 s	0.32	13.62	0.26	12.3
	200	50	06 min09 s	0.52	13.35	0.43	11.89
	100	50	06 min07 s	0.51	12.81	0.43	11.89
	100	25	04 min17 s	0.73	12.79	0.62	11.97
6	200	100	11 min40 s	0.48	24.46	0.43	22.32
	200	50	07 min48 s	0.69	22.77	0.63	21.48
	100	50	06 min56 s	0.78	22.94	0.7	21.99
	100	25	04 min44 s	1.12	22.05	0.99	21.14
11	200	100	14 min40 s	0.56	35.39	0.51	34.28
	200	50	12 min11 s	0.64	33.58	0.59	31.68
	100	50	07 min27 s	1.04	33.79	0.96	32.04
	100	25	05 min32 s	1.37	32.76	1.25	30.73
16	200	100	19 min21 s	0.56	47.56	0.53	45.41
	200	50	18 min37 s	0.55	45	0.53	42.81
	100	50	08 min34 s	1.21	44.55	1.13	42.8
	100	25	07 min05 s	1.42	43.74	1.33	41.96

energy consumption baseline of around 1000 mW when the Jetson client is not processing any task, the hardware extensively consumes around 5000–6000 mW during the Autoencoder phase. Then during the course of training the Transformer-Fourier block, the power consumption of Jetson Nano declines and remains in the range of 4000–5000 mW until the training process is done.

4.3.2.7. Extended experiment. To further examine FATRAF in terms of edge computing performance, we evaluate its resources utilization during the training operation (of both the Autoencoder and Transformer-Fourier block) with varying system parameters, such as: number of features or hidden dimensionality of code sequence d_{hidden}, length of input sequence L_{ae} and code sequence L_{tf}. In terms of the model architecture, we define 2 layers in both blocks to keep the solution light-weight for edge deployment. The same Gas Pipeline data set is applied as above measurements, but only d_{hidden} features is used from the full 16-dimensional training set while the number of samples remains unchanged.

As indicated in Table 7, bandwidth occupation in both upstream and downstream links during the FATRAF operation is insignificant in most cases. With the same dimensionality of input samples, the training time of FATRAF is greatly reduced when decreasing L_{ae} and L_{tf}. In addition, according to Table 8, RAM and average GPU utilization, along with power consumption, seem to escalate to some degree as d_{hidden} grows, considering that GPU is the main computing unit in FATRAF.

5. Discussion and conclusion

In this paper, we have proposed an anomaly detection method that applies the Federated Learning technique to make use of both advantages of distributed learning in different local areas and global updating for all local learning models - FATRAF. FATRAF has been proven to provide high detection performance for time-series data in ICSs in comparison with cutting-edge proposed AD solutions, whilst achieving a remarkable improvement in running time. The reduction of the training time of the learning model down to 1200 s paves the way for this AD solution to be re-trained more frequently during the factory operation.

In the context of ICSs, the normal/abnormal behavior of the system may change over time, due to the aging of devices, changes in system settings, etc., causing the false-positive rate to rise. However, the ability to frequently update the model allows FATRAF to adapt to drifts in the data patterns, thus maintaining the integrity of the anomaly detection system.

Moreover, in the nature of federated learning, after receiving the latest update of the global model, the local model will be improved using locally available data. This will result in automatic adjustment of the current abnormal threshold according to the local situation. With local data, the model will be updated according to the (normal) state of that device within that zone. And that is the benefit of Federated Learning. Therefore, this method can both take advantage of learning from other local data and can adapt the model to best fit

Table 8
RAM, CPU, GPU utilization and Power consumption with different system parameters.

d_{hidden}	L_{ae}	L_{tf}	RAM (%)		CPU (%)		GPU (%)		Power (mW)	
			Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*
1	200	100	65.36	68.01	18.3	42.75	75.16	99	3615.7	4233
	200	50	64.59	67.76	24.83	42.25	63.94	99	3598.93	4233
	100	50	64.15	67.84	24.85	43	67.2	99	3619.27	5048
	100	25	62.83	66.6	32.59	42	55.45	99	3591.44	4387
6	200	100	66.61	69.65	18.81	46	79.68	99	4164.55	5710
	200	50	65.23	68.52	25.01	45	73.35	99	4486.69	5940
	100	50	64.97	68.42	25.8	45.75	66.5	99	3806.39	4387
	100	25	63.78	68.64	33.89	46.25	58.55	99	3902.91	4581
11	200	100	68.85	71.04	17.39	39	81.55	99	4553.91	6080
	200	50	70.26	74.14	20.26	39.25	75.63	99	4814.55	6080
	100	50	65.17	68.87	25.7	46	70.4	99	4226.64	5542
	100	25	63.91	68.19	30.95	44.5	63.8	99	4555.98	5821
16	200	100	74.34	77.67	15.45	47.75	82.56	99	4845.21	6378
	200	50	76.25	80.42	15.49	41.5	82.31	99	5108.66	6734
	100	50	66.48	70.48	24.16	42.25	71.74	99	4518.48	5971
	100	25	66.74	71.44	27.33	41.25	72.8	99	4990.73	6011

the local data before using it in future applications. Of course, there are still many technical challenges and difficulties to overcome to be able to apply federated learning techniques in general in a distributed environment such as: Trade-off between noise and accuracy, System and Statistical heterogeneity, Communication bottlenecks, and Poisoning.

CRediT authorship contribution statement

Truong Thu Huong: Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision. **Ta Phuong Bac:** Methodology, Investigation, Writing – original draft, Writing – review & editing. **Le Anh Quang:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Nguyen Minh Dan:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Le Thanh Cong:** Methodology, Validation, Formal analysis, Writing – original draft, Writing – review & editing. **Nguyen Xuan Hoang:** Methodology, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Formal analysis. **Do Thu Ha:** Methodology, Validation, Formal analysis, Writing – original draft. **Nguyen Tai Hung:** Conceptualization, Methodology, Resources, Writing – original draft, Project administration, Funding acquisition. **Kim Phuc Tran:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Nguyen Tai Hung reports financial support was provided by Hanoi University of Science and Technology. Nguyen Tai Hung reports a relationship with Hanoi University of Science and Technology that includes: employment, funding grants, and non-financial support.

Acknowledgment

This work was supported by Hanoi University of Science and Technology, Hanoi, Vietnam (HUST) under Project T2021-PC-010.

References

Al-Abassi, A., Karimipour, H., Dehghantanha, A., Parizi, R.M., 2020. An ensemble deep learning-based cyber-attack detection in industrial control system. IEEE Access 8, 83965–83973.

Anton, S.D.D, Sinha, S., Schotten, H.D., 2019. Anomaly-based intrusion detection in industrial data with svm and random forests.

Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., Meskin, N., 2020. Cybersecurity for industrial control systems: a survey. Comput. Secur. 89, 101677.

bmon. ⟨https://github.com/tgraf/bmon⟩, Last accessed on May, 2021.

Chang, C.-P., Hsu, W.-C., Liao, I. Anomaly detection for industrial control systems using k-means and convolutional autoencoder. In: Proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). pp. 1–6.

Corallo, A., Lazoi, M., Lezzi, M., 2020. Cybersecurity in the context of industry 4.0: a structured classification of critical assets and business impacts. Comput. Ind. 114, 103165.

Corallo, A., Lazoi, M., Lezzi, M., Luperto, A., 2022. Cybersecurity awareness in the context of the industrial internet of things: a systematic literature review. Comput. Ind. 137, 103614.

Garcia, S., Parmisano, A., Erquiaga, M.J., 2020. IoT-23: a labeled dataset with malicious and benign IoT network traffic. More details here ⟨https://www.stratosphereips.org/datasets-iot23⟩.

He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., Avestimehr, S., 2020. Fedml: a research library and benchmark for federated machine learning. arXiv (2007.13518).

Hil-based augmented ics security dataset. ⟨https://github.com/icsdataset/hai⟩. (Accessed 01 January 2022).

Huong, T.T., Bac, T.P., Long, D.M., Luong, T.D., Dan, N.M., Quang, L.A., Cong, L.T., Thang, B.D., Tran, K.P., 2021. Detecting cyberattacks using anomaly detection in industrial control systems: a federated learning approach. Comput. Ind. 132, 103509.

itrust. Centre for research in cyber security, Singapore University of Technology and Design. ⟨https://itrust.sutd.edu.sg/itrust-labs_datasets/⟩. (Accessed 01 January 2022).

Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T. 2020. SCAFFOLD: stochastic controlled averaging for federated learning. In: D. III, H., Singh, A., (Eds.), Proceedings of the 37th International Conference on, volume 119 of Proceedings of Research, PMLR. pp. 5132–43.

Keogh, E., Lin, J., Fu, A., 2022. Hot sax: efficiently finding the most unusual time series subsequence. In: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05). 8.

Kozik, R., Pawlicki, M., Choraś, M., 2021. A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment. Pattern Anal. Appl. 24.

Kravchik, M., Shabtai, A., 2021. Efficient cyber attack detection in industrial control systems using lightweight neural networks and pca. IEEE Trans. Dependable Secur. Comput (1-1).

Lee-Thorp, J., Ainslie, J., Eckstein, I., Ontanon, S. 2021. FNet: mixing tokens with fourier transforms.

Li, D., Chen, D., Shi, L., Jin, B., Goh, J., Ng, S., 2019. MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks. CoRR (abs/1901.04997).

Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V., 2020. Federated optimization in heterogeneous networks.

Liu, Y., Garg, S., Nie, J., Zhang, Y., Xiong, Z., Kang, J., Hossain, M.S., 2021. Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach. IEEE Internet Things J. 8, 6348–6358.

McMahan, H., Moore, E., Ramage, D., Agüera y Arcas, B., 2016. Federated learning of deep networks using model averaging.

Meng, J., Zhang, Y., Li, Y., Zhao, H., 2022. Spacecraft anomaly detection via transformer reconstruction error.

Mokhtari, S., Abbaspour, A., Yen, K.K., Sargolzaei, A., 2021. A machine learning approach for anomaly detection in industrial control systems based on measurement data. Electronics 10.

Mothukuri, V., Khare, P., Parizi, R.M., Pouriyeh, S., Dehghantanha, A., Srivastava, G., 2021. Federated learning-based anomaly detection for iot security attacks. IEEE Internet Things J (1-1).

Nyc taxi and limousine commission. Last accessed on May, 2021.

Perales Gómez, A.L., Fernández Maimó, L., Huertas Celdrán, A., García Clemente, F.J., 2020. Madics: a methodology for anomaly detection in industrial control systems. Symmetry 12.

Shao, J., Zhong, B., 2003. Last observation carry-forward and last observation analysis. Stat. Med. 22, 2429–2441.

Sheather, S., Marron, J., 1990. Kernel quantile estimators. J. Am. Stat. Assoc. - J. Am. Stat. ASSN 85, 410–416.

Sherstinsky, A., 2020. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Phys. D: Nonlinear Phenom. 404, 132306.

Turnipseed, I.P., 2022. A new scada dataset for intrusion detection research.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I. 2017. Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., (Eds.), Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc.

Xu, C., Shen, J., Du, X., Zhang, F., 2018. An intrusion detection system using a deep neural network with gated recurrent units. IEEE Access 6, 48697–48707.

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., Chen, J., Wang, Z., Qiao, H., 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In: Proceedings of the 2018 Conference, WWW'18, International Conferences Steering Committee, Republic and Canton of Geneva. CHE. pp. 187–96.

Xu, J., Wu, H., Wang, J., Long, M., 2021. Anomaly transformer: time series anomaly detection with association discrepancy.

Zarzycki, K., Ławryńczuk, M., 2021. Lstm and gru neural networks as models of dynamical processes used in predictive control: a comparison of models developed for two chemical reactors. Sensors 21.