Light-Weight Federated Learning-based Anomaly Detection for Time-Series data in Industrial Control Systems

Truong Thu Huong, Ta Phuong Bac, Le Quang Anh, Dan Minh Nguyen, Cong Thanh Le, Hoang Xuan Nguyen, Ha Thu Do, Hung Tai Nguyen, Kim Phuc Tran

Computers in Industry

Light-Weight Federated Learning-based Anomaly Detection for Time-Series data in Industrial Control Systems

--Manuscript Draft--

Manuscript Number:	
Article Type:	Research Paper
Keywords:	Anomaly detection, ICS, Federated Learning, Autoencoder,Transformer, Fourier
Corresponding Author:	Thu Huong Truong, PhD Hanoi University of Science and Technology Hanoi, VIET NAM
First Author:	Thu Huong Truong, PhD
Order of Authors:	Thu Huong Truong, PhD
	Phuong Bac Ta
	Anh Quang Le
	Minh Dan Nguyen
	Thanh Cong Le
	Xuan Hoang Nguyen
	Thu Ha Do
	Tai Hung Nguyen
	Kim Phuc Tran
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.

Research Highlights

- A fast-learning model in 20-minute time scale that can cope with frequent updating
- A light-weight detection scheme in terms of CPU, Memory usage, and running time
- Faster system response upon attacks since detection is implemented near the sources.
- An accurate anomaly detection scheme for time-series data
- Federated learning to reduce bandwidth consumption on the link from Edge to Cloud.

Light-Weight Federated Learning-based Anomaly Detection for Time-Series data in Industrial Control Systems

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

[a] *Hanoi University of Science and Technology, Vietnam*
[b] *School of Electronic Engineering, Soongsil University, Seoul, Korea*
[c] *International Research Institute for Artificial Intelligence and Data Science, Dong A University, Danang, Vietnam*
[d] *Université de Lille, ENSAIT, GEMTEX, F-59000 Lille, France*

Acknowledgment

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

Email addresses: huong.truongthu@hust.edu.vn (Truong Thu Huong), hung.nguyentai@hust.edu.vn (Nguyen Tai Hung)

Light-Weight Federated Learning-based Anomaly Detection for Time-Series data in Industrial Control Systems

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.

Keywords: Anomaly detection, ICS, Federated Learning, Autoencoder, Transformer, Fourier

1. Introduction

Industrial control systems (ICS) [1] 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. Since an ICS architecture contains valuable information that can affect the performance of the whole industry, it becomes a

significant target for attacks from a variety of threats. 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 [2], LSTM [2], and GRU [3]. 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.

To tackle those limitations, we propose to use a Transformer-based approach for anomaly detection in this paper. Unlike the other existing sequence-to-sequence models, Transformer, which was initiated in [4] to solve NLP problems, 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 [5]. 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 computa-

tional 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 combination of Autoencoder - Transformer - Fourier to bring 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) [6] 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.

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. Solutions have been proposed to detect cyber attacks for ICS, such as [7], [8], [9],[10], and [11]. In [7], 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 [12] and gas pipeline data sets [13], 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 [8] proposed a measurement intrusion detection system approach to detect any common abnormal activity in an ICS system with the HAI data set [14]. They applied the well-known supervised learning algorithms such as K-Nearest Neighbour, 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 [8] 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 [8], 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 [10]. 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 [9] 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 [10], 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 [15], [16], [17], and [18], for example, did not yield significant anomaly detection performance when employing LSTM and GRU for ICS time series data. In work [15], 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 [16] 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 [17] 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 [18], 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 [18] did not account for the model training time, which is the major drawback of LSTM and GRU. Work [19] proposed a cyberattacks detection mechanism using the combination of Variational Autoencoder (VAE) and LSTM. Although the detection performance is considerably high, the model in [19] faces difficulty in terms of running time since the LSTM block requires a long training time. Liu *et al.* [20] 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 [4] 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 [21] 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 [22] 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 [23] brings remarkable performance. By leveraging a transformer encoder followed by a two-layer feed-forward neural network, the classifier deployed in [23] performs well on the Aposemat IoT-23 data set [24] 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 [23], 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. Besides, to the best of our knowledge on the relevant studies, our transformer-based approach is considered at the forefront of anomaly detection deployment for ICS ecosystems.

3. System Architecture Design

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 Figure 1, FATRAF comprises two main components:

- **Edge sites:** In a factory, 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 an edge device in the vicinity of the corresponding manufacturing zone. Designated to monitor anomalies, these edge devices employ the local data as inputs for training its own local anomaly detection model - ATRAF (**A**E-**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 differ-

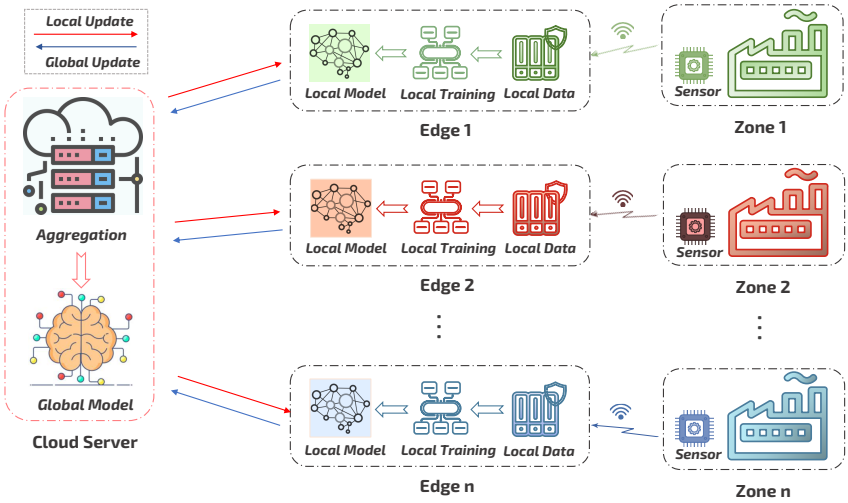


Figure 1: FATRAF Architecture

ent edges. The whole process of model aggregation and global model updating down to all local models are called Federated Learning.

As Figure 2 depicts, the Federated learning process of FATRAF comprises the following key steps:

1. *System Initialization:* At the beginning, the cloud server establishes a global model with specific learning parameters (i.e., the global FATRAF model), and sending it to each edge device of each corresponding edge site.
2. *Local Training:* After receiving that initial configuration, by utilizing the on-site data collected from sensors, edge devices conduct a local training process. Accordingly, anomaly detection is deployed right at these edge sites, enabling fast and timely system response upon attacks.
3. *Local Model Update:* After the local training, the edge devices send back the learned weights w_{t+1}^k to the cloud server for aggregation.
4. *Model Aggregation:* After deriving all trained weights w_{t+1}^k, the cloud server federates them and constructs a new global model version by the formula proposed by [25], as follows:

$$w_{t+1} = \sum_{k=1}^{n} \frac{v_k}{v} w_{t+1}^k \tag{1}$$

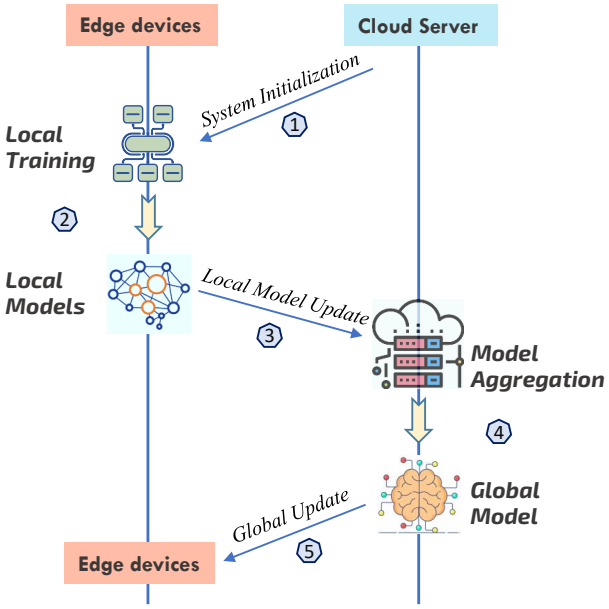


Figure 2: Flow diagram of FATRAF

Where:
n: the number of edge sites
v_k: amount of data of the k^{th} edge site
v: total amount of data of all edge sites
w_{t+1}^k: weight of the local model at the k^{th} edge site at time $t+1$
w_{t+1}: weight of the federated global model 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 [26, 27], through our various experiments, the approach of federating local weights presented in Equation1 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 so as to update the local models. In the next learning rounds, this communication process will repeat from the second step in order to optimize the local models until the global learning model converges.

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 Auto Encoder (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 classifiers such as the recent transformer-based work [23] for IoT applications.

As illustrated in Figure 3, the AE model 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. It can be seen in Figure 3, we use the Autoencoder for the local training process and only the encoder part for the testing process. As the output of the AE Encoder block, 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.

Aiming to achieve high anomaly detection performance for time-series data and better running time, Transformer is used as a remedy for other

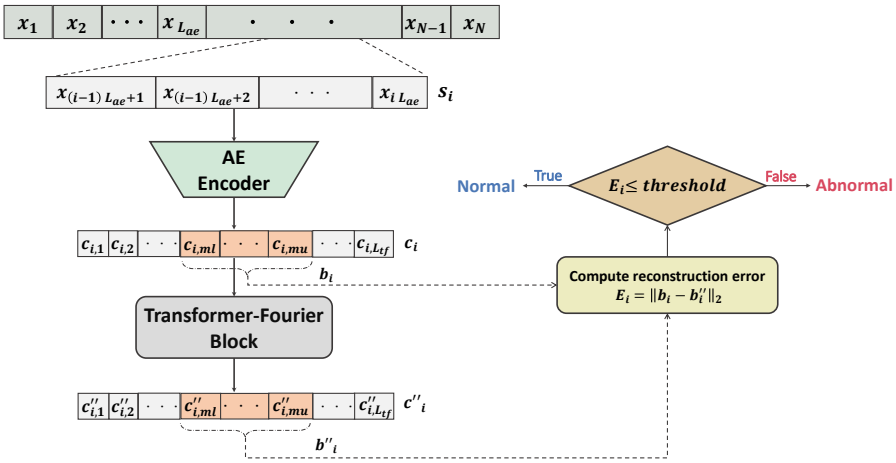


Figure 3: ATRAF Learning Model

time-series approaches such as LSTM and GRU since the Transformer model processes sequences in parallel. In the Transformer-Fourier block, the training process is significantly accelerated by comparing to, for example, that of 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. The Transformer-Fourier block consists of a Transformer encoder layer followed by a Fourier layer (specifically a Fourier Transform with a position-wise feed-forward network). 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. 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.

$$\overline{X} = Decoder(Encoder(X))$$

Where:
X and $\overline{X}$ are the input and output of AE

- Input $X = \{x_1, x_2, \ldots, x_N\}$, where $X \subset \mathbb{R}^D$, is divided into k non-overlapping sequences $\{s_1, s_2, \ldots s_k\}$ where $s_i = \{x_{(i-1)L_{ae}+1}, \ldots, x_{iL_{ae}}\}$ is a sequence of length L_{ae}.
- These sequences are then used to train the local AE model. After the training process, the sequences are fed through the AE model's encoder, producing compressed code sequences $\{c_1, c_2, \ldots, c_k\}$ where $c_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,L_{tf}}\}$ with $L_{tf} \leq L_{ae}$.

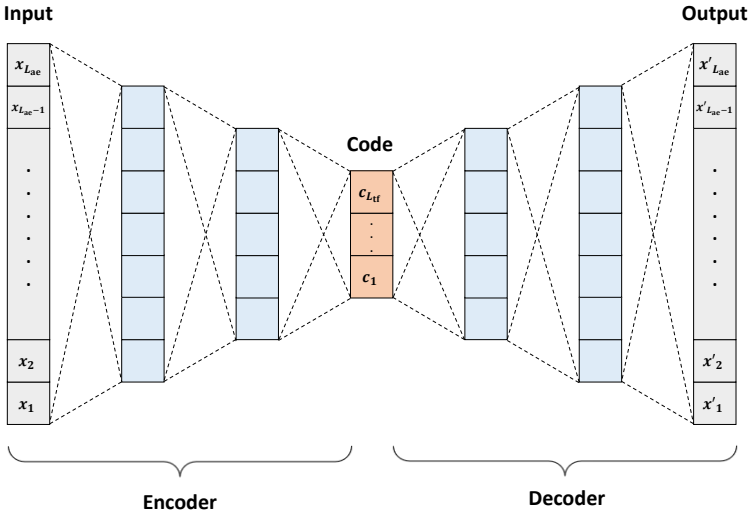


Figure 4: Autoencoder Configuration

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 $\frac{L_{ae}}{L_{tf}}$.

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 only 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 speeds, especially at long input lengths [5].

The hybrid model of Transformer and Fourier transform mixing sublayer can 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. The operations of the Transformer-Fourier block in Figure 5 can be described as follows:

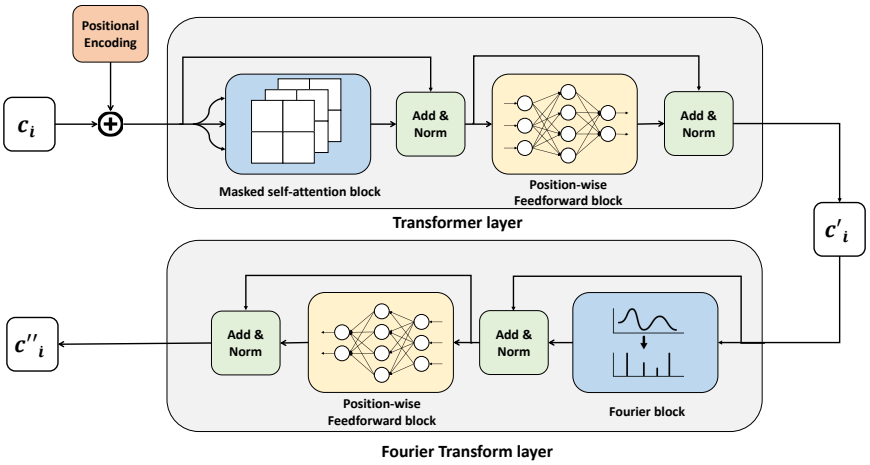


Figure 5: The Transformer-Fourier block

- Each output code sequence c_i of the local AE models are 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 $c_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,L_{tf}}\}$.

$$PE_{p,2i} = \sin(\frac{p}{10^{\frac{8i}{d_{hidden}}}}) \tag{2}$$

$$PE_{p,2i+1} = \cos(\frac{p}{10^{\frac{8i}{d_{hidden}}}}) \tag{3}$$

  where p is the position, i is the dimension index and d_{hidden} is the size of the code sequences' hidden dimension.

- A contiguous sub-sequence from index ml to mu (as illustrated in Figure 3) in each resultant sequence is blocked from the Transformer encoder (by assigning the respective score in the self-attention matrix

to minus infinity). The Transformer encoder block 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 $c'_i = \{c'_{i,1}, c'_{i,2}, \dots c'_{i,L_{tf}}\}$ of the same dimensionality as the input sequence c_i.

- This embedding c'_i is then fed into the Fourier Transform layer. The Self-Attention sublayer of the Transformer encoder is replaced with an unparameterized Fourier sublayer. The Fourier sublayer applies two 1-dimensional Discrete Fourier Transforms along the hidden dimension and the sequence dimension of the sublayer's input and takes the real part of the result, i.e., $\Re(\mathscr{F}_{hidden}(\mathscr{F}_{sequence}(c'_i))$. The unparameterized Fourier transform is a relatively effective and light-weight mixing method and is able to retain 92-97% the performance of a Transformer encoder while only taking 20% the training time [5]. The output of the Fourier transform layer is a reconstructed sequence of the input sequence c_i which is denoted $c''_i = \{c''_{i,1}, c''_{i,2}, \dots c''_{i,L_{tf}}\}$. The reconstruction errors are calculated between the masked sub-sequence of c_i, i.e., $b_i = \{c_{i,ml}, \dots, c_{i,mu}\}$ and the corresponding reconstructed sub-sequence $b''_i = \{c''_{i,ml}, \dots, c''_{i,mu}\}$ as demonstrated in Figure 3.

$$E_i = \|b_i - b''_i\|_2 \tag{4}$$

- An anomaly threshold λ_{Th} is determined by the Kernel Quantile Estimation (KQE) technique [6] that is used on the reconstruction errors to classify the input sequence $s_i = \{x_{(i-1)L_{ae}+1}, \dots, x_{iL_{ae}}\}$. Without loss of generality, assume that $E_1 \leq E_2 \leq \dots \leq E_k$. The anomaly threshold is determined by

$$\lambda_{Th} = \sum_{i=1}^{k} \left[\int_{\frac{i-1}{k}}^{\frac{i}{k}} \frac{1}{h} K(\frac{t-q}{h}) dt \right] E_i \tag{5}$$

Where K is the density function, chosen as the standard Gaussian kernel $K(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})$; the asymptotically optimized bandwidth $h = \sqrt{\frac{p(1-q)}{k+1}}$ controls the estimator's smoothness, and q $(0 < q < 1)$ is the preset value.

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 this study, 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 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.

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 [30], 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.

Finally, all data sets should be normalized to have all processed features to be in the same scale between 0 and 1 according to the min-max normalization described in Equation 6:

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{6}$$

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.

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

Data sets	Description
Gas Pipeline [13]	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 [12]	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 [14]	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 [28]	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 [28]	Time-series of the heartbeat electrical signals
Respiration [28]	Measurements of patient's respiration when waking up by thorax extension
Gesture [28]	2-feature data set indicates the right hand's coordinates while performing different actions
Space shuttle [28]	Solenoid current's measurements of a Marotta MPV-41 series valve cycled on and off
NYC taxi [29]	Information on New York taxi passenger data stream (Jul 2014 - Jun 2015)

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 i7-10700K CPU with 16GB RAM.

In Federated-learning mode, the expriment 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, 64GB 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* [31] is used to measure bandwidth occupation in each link between the Edge–Server link in the Federated Learning–based architecture.
- Built-in utility *tegrastats* keeps track of computational resources usage as well as energy consumption on edge devices during 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 [32] 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 [33], 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.

Experiment 1 - Federated Learning vs. Centralized learning

At first, in this experiment, we will compare the detection performance of FATRAF with its centralized computing mode (i.e. implementing the ATRAF learning model in the centralized cloud mode). Table 3 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 3 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.

Table 3: 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

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 [19] - Federated Learning called FL-VAE-LSTM. As Table 4 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. 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 [19] is also compared to show that the training time of FATRAF much outperforms FL-VAE-LSTM.

Table 4: Federated-Learning approaches over the time-series data sets

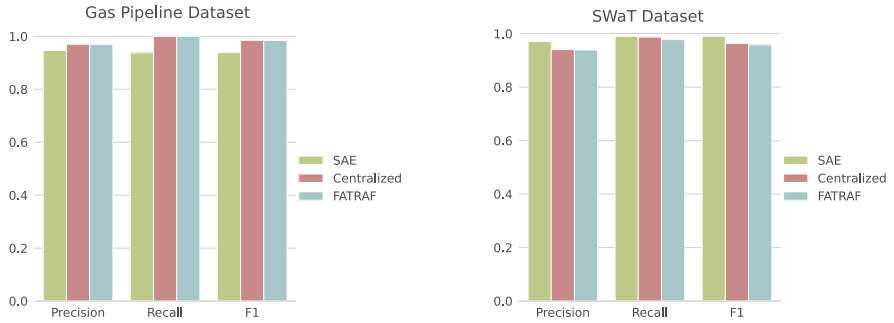
Data set	FL-VAE-LSTM [19]			FATRAF		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Power Demand	0.7355	0.9100	0.8135	0.9285	0.9442	0.9363
Gas Pipeline	0.9609	0.9982	0.9792	0.9683	1	0.9839
Space shuttle - TEK14	0.8623	0.8431	0.8536	0.9738	1	0.9867
Space shuttle - TEK16	1	1	1	0.9296	1	0.9635
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

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 [7, 8, 22] in the field of anomaly detection in ICS.

Figure 6 depicts the detection performance of the 3 solutions: SAE proposed in [7], our FATRAF and its centralized computing mode on 2 different data sets: Gas Pipeline [13] and SWaT [12]. As these data sets are primary case studies in [7], 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.



(a) Gas Pipeline data set

(b) SWaT data set

Figure 6: Detection performance of SAE [7] vs. FATRAF vs. ATRAF's centralized mode on two datasets

Another comparison is made between our proposed solution and the measurement intrusion detection system (MIDS) introduced in [8], which utilizes 3 machine learning techniques such as KNN, Decision Tree, and Random Forest. As can clearly be seen from Table 5, 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 [8] presented much higher classification performance, approximately ideal results, than our shown experimental results. The reason could be that those performance metrics may have been 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.

Table 5: FATRAF and ATRAF's centralized mode vs. Measurement Intrusion Detection System [8]

Data set		HAI			Gas Pipeline		
		Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
MIDS [8]	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

Finally, Figure 7 shows the detection performance differences of proposed solutions and Anomaly Transformer [22] 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.

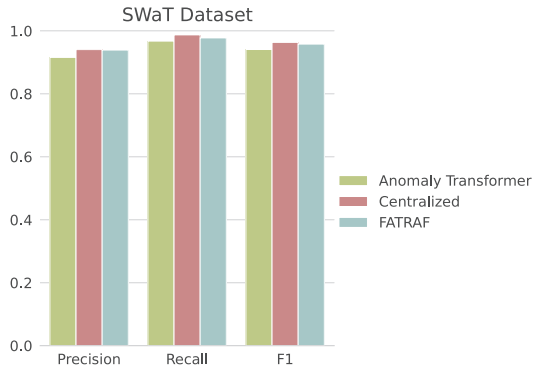


Figure 7: FATRAF and ATRAF's centralized mode vs. Anomaly Transformer [22]

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.

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. Figure 8 demonstrates the memory usage of an NVIDIA Jetson Nano representing an edge device during the whole operation of 1155 seconds (roughly 19 minutes). As it can be noticed, the autoencoder phase takes about 410 seconds or nearly 7 minutes, 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 4GB 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.

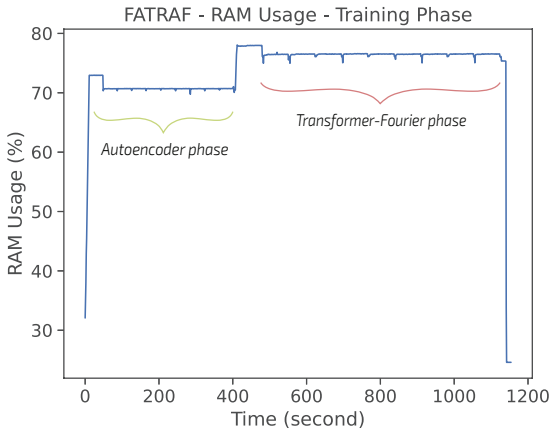


Figure 8: Memory Usage of an Edge device (NVIDIA Jetson Nano) during the FATRAF Training Phase

CPU usage

In order to see how CPU is occupied for the training phase of the learning model, and to analyze how light-weight our proposed solution is, CPU usage is measured and presented in Figure 9. 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.

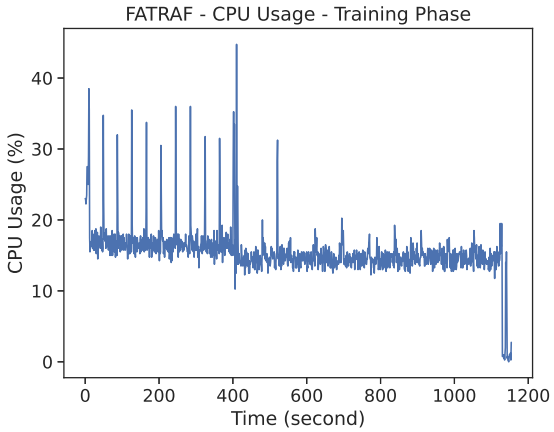


Figure 9: CPU Usage of an Edge device (NVIDIA Jetson Nano) during the FATRAF Training Phase

GPU Usage

The GPU usage of the edge hardware based on NVIDIA Jetson Nano is illustrated in Figure 10. 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.

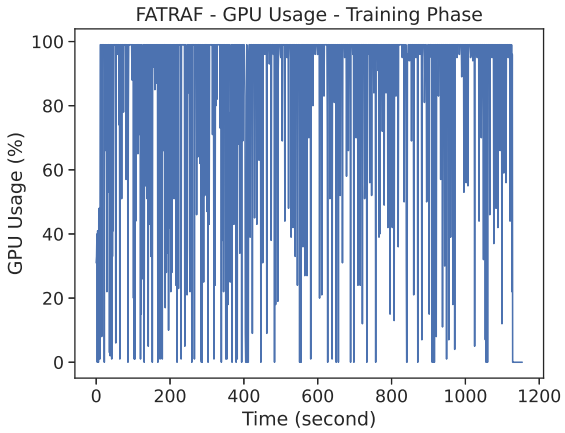


Figure 10: GPU Usage of NVIDIA Jetson Nano during FATRAF Training Phase

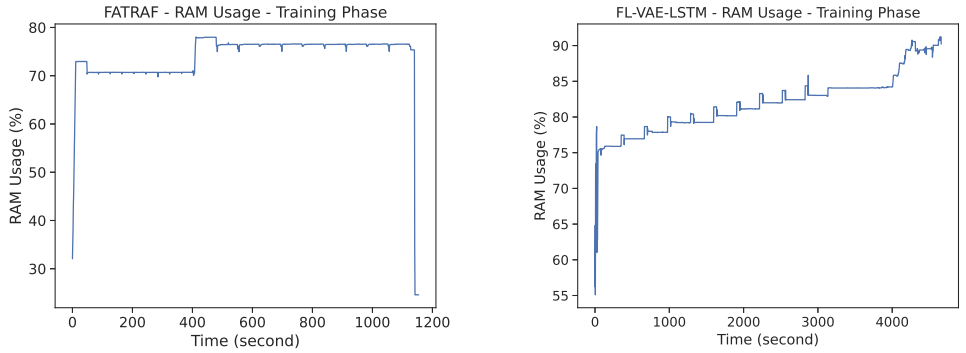
Running time evaluation

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 [19] 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 Figure 11 illustrates, when training the 2 models on the same hardware, the training time of the hybrid of AE - Transformer - Fourier takes 1200 seconds overall, whilst the combination of VAE - LSTM needs around 5000 seconds to finish the training process.

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.



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

Figure 11: Memory Usage and Training time of FATRAF vs. FL-VAE-LSTM [19]

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, Figure 12 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.

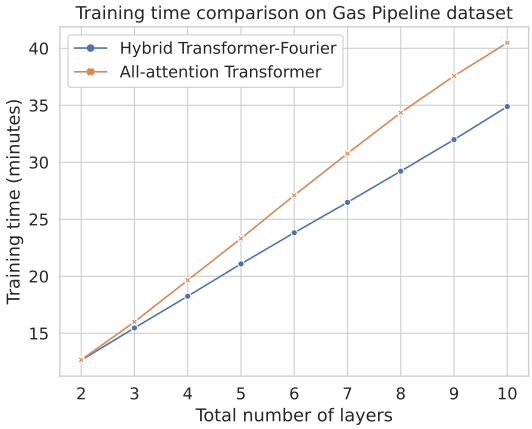


Figure 12: Training time of Hybrid Transformer-Fourier model vs. All-Attention-Sublayer Transformer model

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. Figure 13 and Figure 14 depict the bandwidth occupied in the upstream 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 Figure 13 and Figure 14, there are 10 spikes of bandwidth occupation of around 45 KiB/s. Each spike spans no more than 2 seconds 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.

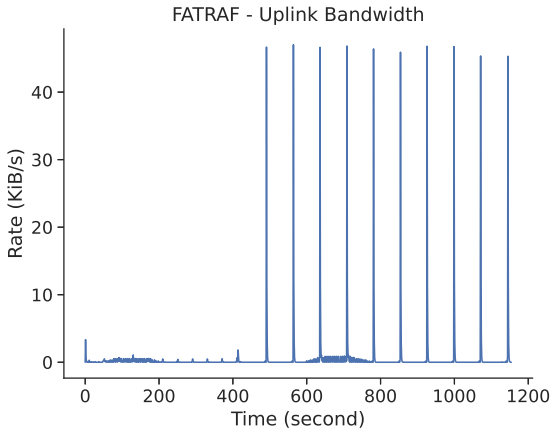


Figure 13: Bandwidth occupation in the Edge-Cloud upstream link of FATRAF

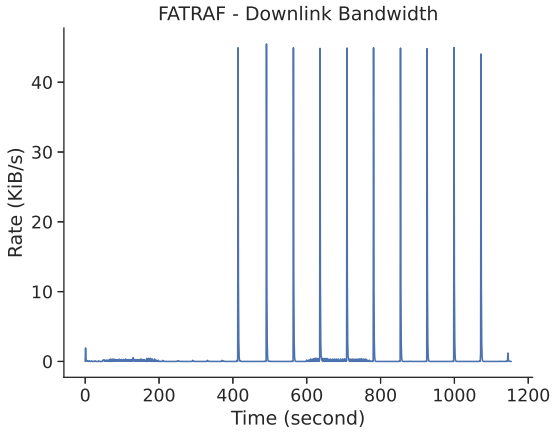


Figure 14: Bandwidth occupation in the Edge-Cloud downstream link of FATRAF

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. Figure 15 shows a sketch of how FATRAF consumes power during each phase in

the whole training duration. With an 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.

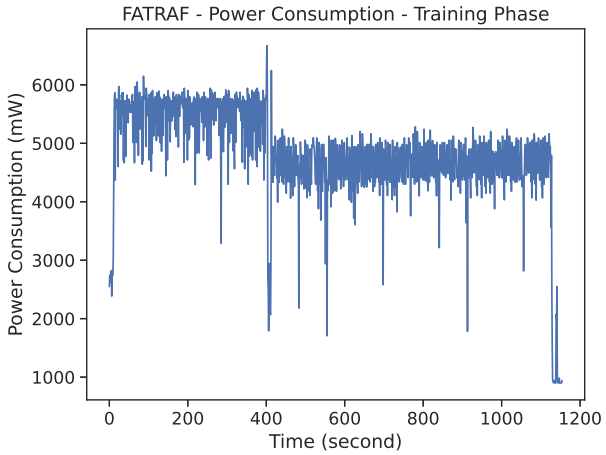


Figure 15: Power Consumption of NVIDIA Jetson Nano during the FATRAF Training Phase

5. 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 seconds paves the way for this AD solution to be re-trained more frequently during the factory operation. In turn, it helps the security architecture in ICS to be able to frequently update changes in normal behaviors of the smart devices installed in a smart factory 4.0.

References

[1] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, N. Meskin, Cybersecurity for industrial control systems: A survey, Computers & Security 89 (2020) 101677.

[2] A. Sherstinsky, Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, Physica D: Nonlinear Phenomena 404 (2020) 132306.

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

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

[5] J. Lee-Thorp, J. Ainslie, I. Eckstein, S. Ontanon, FNet: Mixing tokens with fourier transforms (2021).

[6] S. Sheather, J. Marron, Kernel quantile estimators, Journal of The American Statistical Association - J AMER STATIST ASSN 85 (1990) 410–416.

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

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

[9] 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) 1–1.

[10] C.-P. Chang, W.-C. Hsu, I. Liao, Anomaly detection for industrial control systems using k-means and convolutional autoencoder, in: 2019

International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–6.

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

[12] itrust, centre for research in cyber security, singapore university of technology and design, https://itrust.sutd.edu.sg/itrust-labs_datasets/, Accessed: 2022-01-01.

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

[14] Hil-based augmented ics security dataset, https://github.com/icsdataset/hai, Accessed: 2022-01-01.

[15] K. Zarzycki, M. Ławryńczuk, 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 (2021).

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

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

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

[19] T. T. Huong, T. P. Bac, D. M. Long, T. D. Luong, N. M. Dan, L. A. Quang, L. T. Cong, B. D. Thang, K. P. Tran, Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach, Computers in Industry 132 (2021) 103509.

[20] 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 communication-efficient on-device federated learning approach, IEEE Internet of Things Journal 8 (2021) 6348–6358.

[21] H. Meng, Y. Zhang, Y. Li, H. Zhao, Spacecraft anomaly detection via transformer reconstruction error.

[22] J. Xu, H. Wu, J. Wang, M. Long, Anomaly transformer: Time series anomaly detection with association discrepancy, 2021.

[23] R. Kozik, M. Pawlicki, M. Choraś, A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment, Pattern Analysis and Applications 24 (2021).

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

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

[26] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A. T. Suresh, SCAFFOLD: Stochastic controlled averaging for federated learning, in: H. D. III, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 5132–5143.

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

[28] 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), pp. 8 pp.–.

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

[30] J. Shao, B. Zhong, Last observation carry-forward and last observation analysis, Statistics in Medicine 22 (2003) 2429–2441.

[31] bmon, `https://github.com/tgraf/bmon`, Last accessed on May, 2021.

[32] 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 (2020).

[33] 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.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:
