

Enhancing Self-Adaptive Cyber-Physical Systems using Federated Machine Learning

Nabila Azeri¹, Ouided Hioual² and Ouassila Hioual^{2, 3}

¹ ICOSI Laboratory, Abbes Laghrour University, Khenchela, Algeria

² Abbes Laghrour University, Khenchela, Algeria

³ LIRE Laboratory, Constantine 2 University, Algeria

Abstract

Cyber-physical systems (CPS) seamlessly integrate physical elements with computing and communication technologies for process monitoring and control. These systems, operating in dynamic environments, face challenges due to shifting user requirements, device dynamics, and environmental fluctuations, potentially impacting service quality. The development of adaptive CPS becomes essential to respond effectively to these challenges, requiring dynamic reconfiguration, resource optimization, and intelligent responses. While numerous self-adaptation approaches exist, most rely on Machine Learning (ML) techniques, emphasizing performance gains but often neglecting security and privacy concerns tied to centralized data processing.

To harness ML's potential in CPS comprehensively, we advocate a holistic approach that prioritizes security, data privacy, and adaptability. In this paper, we introduce an innovative approach to adaptive CPS development, leveraging Federated Machine Learning (FML) technology. This approach reconciles adaptability with data security and privacy.

Keywords

Cyber-physical systems, Federated Machine Learning, Multi-layer architecture, TensorFlow Federated

1. Introduction

Cyber-physical systems (CPS) are complex systems that integrate physical components with computing and communication technologies to monitor and control physical processes [1]. These systems are ubiquitous in various fields, including transportation, healthcare, manufacturing, and energy [2]. Nowadays, CPS have evolved from stand-alone computers to highly dynamic and complex systems with ability to synergize real-time data processing, control, and communication to facilitate seamless interactions between the digital and physical worlds. As CPS grow in complexity and functionality, they become instrumental in enhancing efficiency, optimizing resource utilization, and enabling intelligent decision-making. However, this increasing complexity brings forth a challenge: how to ensure that these systems not only meet their design requirements but also adapt and evolve in response to rapid environmental change.

The need for adaptation within CPS stems from their inherently intricate nature [3]. Unlike traditional systems, CPS exhibit a high degree of heterogeneity, involving diverse components that operate at different levels of abstraction. Moreover, the dynamic and often unpredictable nature of the physical world adds an additional layer of complexity. As a result, the design and development of CPS that can effectively navigate this complexity and remain in compliance with requirements have become a principal concern [4, 5].


In the face of these challenges, a paradigm shift is required in how we conceptualize the design and operation of CPS. The conventional approach of designing systems with static, pre-defined solutions is no longer sufficient. Instead, there is a growing demand for self-adapting CPS, which is systems that can autonomously monitor their environment, assess the perturbations or

TACC 2023: Tunisian-Algerian Joint Conference on Applied Computing, November 6 - 8, Sousse, Tunisia

✉ azeri.nabila@gmail.com (N. Azeri); hioual.ouided@univ-khenchela.dz (O. Hioual); hioual_ouassila@univ-khenchela.dz (O. Hioual)



© 2023 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

variations occurring within it, and dynamically adjust their behavior to ensure continued functionality and compliance with performance criteria.

To support developing such new functionalities, Machine Learning (ML) algorithms were merged. These latter have achieved impressive results providing solutions to practical large-scale problems. ML offers a powerful set of approaches, techniques, and predictive models that can process a large amount of information and help solve many use cases efficiently [6-8].

Numerous studies have explored the utilization of ML techniques to develop flexible and adaptive CPS capable of responding to changing environmental conditions. These techniques often involve a central storage of data, followed by the training of a model on this centralized data repository. This approach facilitates comprehensive analysis and decision-making on a global scale. However, this centralized approach gives rise to potential challenges concerning data privacy and security, as all the information is consolidated in a single location. This situation calls for the emergence of Federated Machine Learning (FML), a methodology that embraces a decentralized perspective while retaining data on local devices or servers, such as mobile phones or IoT devices. In this paradigm, the model is disseminated and trained locally on these distributed datasets, eliminating the necessity of transferring data to a central repository [9].

This research introduces an innovative approach for the development of self-adaptive CPS. The approach we propose is based on the FML paradigm, which gives CPS the ability to dynamically adjust their behaviors in real time, while ensuring data privacy and security. Our approach has been validated through a real-world case study focused on predicting faults in industrial CP

The rest of the paper is structured as follows. Section 2 presents some related work. Section 3 introduces our previous work concerning a centralized architecture for CPS. Section 4 describes the proposed FML-based architecture. The implementation and application of our architecture on a real-world example are detailed in Section 5. Finally, some concluding remarks and directions for future work are given in Section 6.

2. Related Work

In recent years, the implementation of CPS using ML has gained substantial attention in the research community. This section explores some works in the domain of CPS implementation, highlighting the integration of ML techniques. These works leverage centralized data processing and model training, pooling data from various sources to build global models. For instance, in Authors in [10] propose an innovative approach that harnesses the power of ML to enhance the security and reliability of Medical Cyber-Physical Systems (MCPS). In their work, they introduce the Improved Wireless Medical Cyber-Physical System, a framework that leverages ML techniques to protect patient health data and ensure data integrity in wireless medical environments. At the core of this framework is the use of ML, specifically deep neural networks, for attack detection and classification. The authors recognize that the heterogeneity of devices in MCPS, including mobile devices and body sensor nodes, poses security vulnerabilities. To mitigate these threats, they employ ML as a robust security solution.

In [11], authors propose a complete real-world CPS implementation cycle, ranging from machine data acquisition to processing and interpretation. In fact, they propose a CPS for machine component knowledge discovery based on clustering algorithms using real data from a machining process. More precisely, they propose the development of a component behavior multidimensional pattern using machine learning clustering techniques. Clustering algorithms aim to partition a centralized dataset into clusters, interpreted as behavior patterns. Some properties of these clusters like the mean can give insight into their interpretation.

In [12], physical data machine learning approaches are developed and integrated for detecting cyber physical attacks in cyber manufacturing system. Authors developed two examples with simulation and experimentation to test and demonstrate the physical data machine learning security approach. Three different machine learning algorithms are

implemented with image classification. The anomaly detection method returned the highest accuracy of 96.1% in detecting a malicious defect in printing process.

Authors in [13] provide a self-adaptive and scalable prediction/detection mechanism for CPS. They propose a framework called AAPF-CPS, which combines several machine learning algorithms with statistical tests. With multiple classification algorithms, AAPF-CPS analyzes CPS network logs simultaneously and in real-time. Friedman's test is also used to rank each classifier for each context in AAPF-CPS.

In [14], authors present an approach for anomaly detection regarding predictive maintenance in a CPS. In particular, they are focusing on historical sensor data from a real reflow oven that is used for soldering surface mount electronic components to printed circuit boards. The sensor data comprises information about the heat and the power consumption of individual fans inside a reflow oven. The data set contains time annotated sensor measurements in combination with additional process information over a period of more than seven years.

Authors in [15] propose a novel approach to address the challenges associated with implementing CPS in Industry 4.0. To tackle this, the study introduces an integrative machine-learning method aimed at reducing computational complexity and improving CPS applicability as a virtual subsystem. The approach leverages the power of the Random Forest algorithm and a time-series deep-learning model based on Long Short-Term Memory networks. This combination enables real-time monitoring and facilitates faster corrective adjustments for machines within the CPS environment. A key innovation in this method is the early fault detection mechanism, which triggers an alarm well before a machine failure. This proactive approach empowers shop-floor engineers to make necessary adjustments or perform maintenance, mitigating the impact of machine shutdown.

Centralized ML solutions, while effective in optimizing CPS operations, inherently concentrate sensitive data in a single location. This concentration poses significant risks, such as potential data breaches, and raises concerns about user privacy. Additionally, centralization can introduce single points of failure, compromising system reliability, especially in scenarios where data transmission or the central processing unit becomes concealed. By embracing a decentralized solution like FML, we aim in this paper to enhance adaptability while addressing critical concerns surrounding data security and privacy in CPS, enabling a safer and more resilient future for these systems.

In the next section, we will give and explain briefly our earlier research contributions. By examining these pre-existing contributions, we can better explain the evolutionary way that has led us to our current approach.

3. A Multi-layer Architecture for CPS

CPS are characterized by their distribution and loose coupling of both cyber systems and physical systems, all of which are monitored and controlled according to user-defined semantic laws [3]. It is important to define the architecture of this system as well as its components and their functionalities besides the used technology. The general objective of a CPS is to collect multiple data from various sources such as the data received from the sensors installed in the different production machines. Thereafter, we convert them into information in the cyber world in order to process, to understand them, and then turn them into appropriate actions in the physical world. Therefore, our architecture presented in [16] contains the main software modules that will ensure CPS functionalities such as: resource/data management, process planning, and the different modules which convert machines to be self-aware, self-learning and self-reconfiguring. As shown in Figure 1, the proposed architecture consists of the following layers:

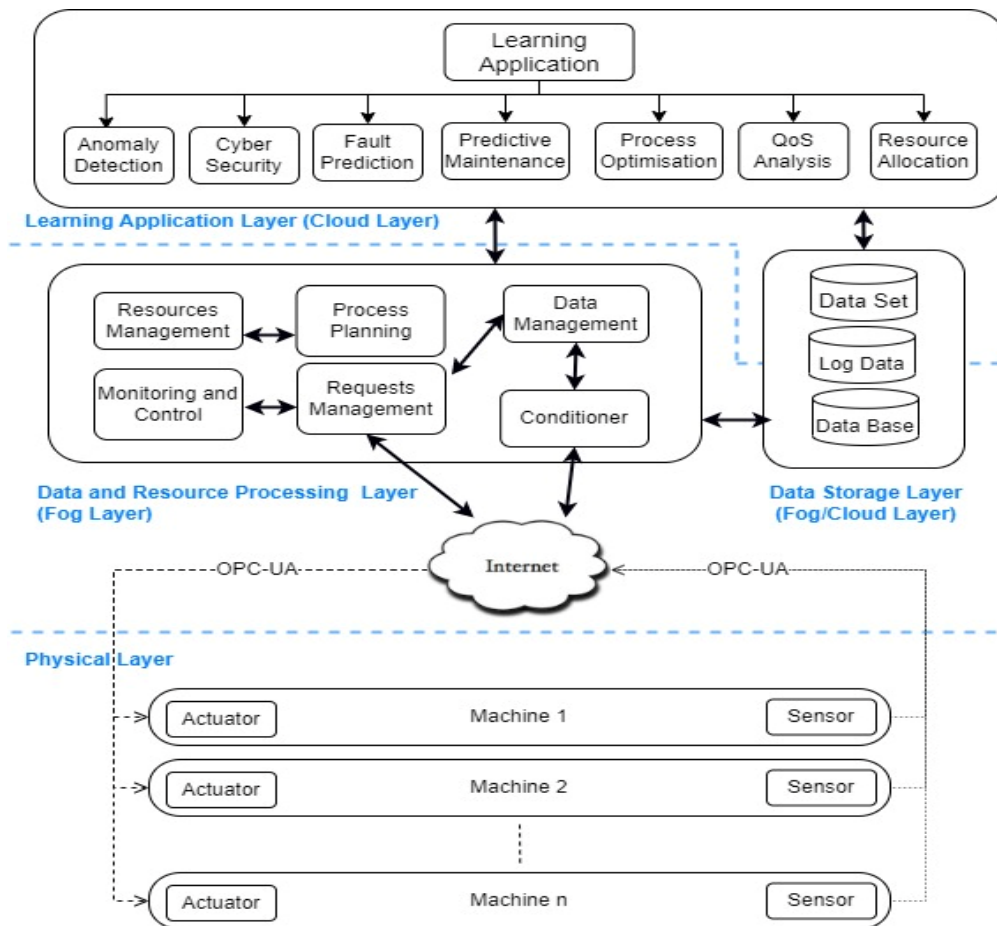


Figure 1. The proposed architecture [16]

Physical layer: It represents the basic local assembly of machines connected to the Internet via the OPC-UA (Open Platform Communications Unified Architecture) protocol to ensure communication standardization between different units. This layer consists mainly of sensors and actuators. Sensors are used to collect signals from machines and then transform them. Whereas, the actuators take electrical signals and combine them with a source of energy to create a physical movement.

Data/Resource Processing Layer: This layer collects data from various sources, and then it interacts directly with the data producer at the physical layer and the data storage in the Fog/Cloud. It contains the following modules: the conditioner, the data management, the resource management, the planning process, the monitoring and control module and the request management module.

Data Storage Layer: This layer serves as the storage base for the CPS and it is distributed between the Fog and the Cloud, and this depends on the nature of the data and the time required to process it.

Learning Application Layer: This layer ensures different functionalities such as: Resource allocation, QoS analysis, Process optimisation, Predictive maintenance and Fault detection. Within this layer, a variety of ML techniques, including regression, classification, clustering, and reinforcement learning, take center stage. These models are meticulously trained on a rich dataset, combining historical knowledge with real-time information, culminating in the ability to make predictions and informed decisions that drive the system's adaptive capabilities.

To demonstrate the viability and effectiveness of this layer, we have implemented it in a practical context. This application aligns with the realm of : Failure Prediction Using Supervised and Unsupervised Learning Algorithms [16]. In this real-world scenario, our system leveraged

the power of supervised and unsupervised learning algorithms to predict failures in a dynamic environment such as CPS.

4. Towards a FML-Based architecture for CPS

This section provides an overview of the proposed FML-based architecture, focusing on the components of the architecture and how they interact with each other. We will only present what we have added compared to our previous architecture in terms of FML. As shown in Figure 2, the architecture consists of three main entities: local collectors, local aggregators and a central server.

4.1. Architecture Components

Local collectors: Local Collectors are responsible for collecting and managing data from the local devices or nodes within a particular group or federation. They can encompass a wide range of devices, including industrial equipment, individual smartphones, IoT devices, sensors, and other endpoints. The Local Collectors ensure that data remains secure and private, often by aggregating or summarizing the data before it's sent to the aggregator. The primary focus of Local Collectors is to prepare and send relevant information to the aggregator, respecting privacy and security constraints.

Local Aggregators: Local Aggregators are responsible for aggregating the locally computed model updates from devices or nodes within their own federation. Each device trains its model locally using its own data, and then these local models are sent to the Local Aggregator. The Local Aggregator combines the models in some way, often by averaging the model parameters, to generate a global model update. Then, this global model update is shared with other federated groups or used to update the global model maintained by the server. Local aggregators facilitate collaborative learning across devices while ensuring that sensitive data remains on the local devices.

Central Server: The Central Server has a crucial role in the FML framework. Its main goal is to facilitate the aggregation of knowledge that originates from various local aggregators. Due to privacy considerations, our architecture shares only the learned data that is coordinated by the central server, while the private or sensitive data will stay at the local aggregators. Due to potential variations in data types and machine learning models across local aggregators, the Models Manager module in our architecture assumes the role of overseeing and harmonizing these diverse models, ensuring effective management and coordination. The updating of the global model is achieved through a synchronization process that involves both the global model and the local aggregators. This synchronization is achieved by the Models Aggregation module, which manages the coordination between these components.

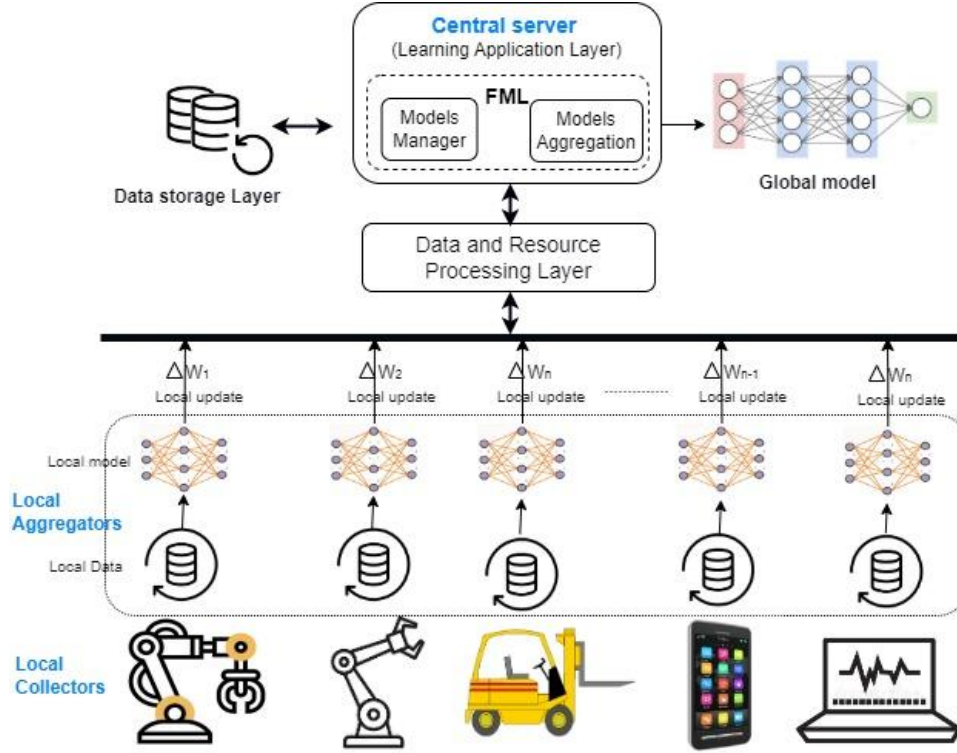


Figure 2. The proposed FML-based architecture

4.2. Mode of Operation

In our architecture, the goal is to collaboratively train a global model using data from N clients, each having their own local dataset denoted as D_i . This dataset consists of a set of K_i labeled pairs:

$$D_i = \{X_i^{k_i}, Y_i^{k_i}\}_{k=1}^{k_i} \quad (1)$$

Where $X_i^{k_i}$ is referred to the training input, and $Y_i^{k_i}$ is the corresponding label.

In our architecture, the process revolves around collaborative training between clients and a central server. The goal is to update a global model by utilizing the local models trained on each client's local data. Algorithm 1 captures the behavior of each client in our architecture.

Algorithm 1 : Client Side

Input : M_0 : initial global model

D : Local data

Output : the gradients of the local loss

- 1 $M_{local} \leftarrow M_0$
 - 2 $M_{local} \leftarrow \text{TrainLocalModel}(M_{local}, D)$
 - 3 $\text{gradientsLocal} \leftarrow \text{ComputeGradients}(M_{local}, D)$
 - 4 $\text{SendToServer}(\text{gradientsLocal})$
-

The client-side process begins with the initialization of a local model (M_{local}) using the initial global model (M_0). This local model is provided by the central server and it serves as the starting point for the client's training process (line 1). Subsequently, the local model is trained using the client's local data (line 2). Through this local training, a client trains its local model M_{local} using its local dataset D to minimize a local loss function : $L_i(M_{local})$. This is done by the formula:

$$M_{local} = \underset{M}{\text{argmin}} L_i(M_0) \quad (2)$$

Following the training phase, the gradients of the local loss with respect to the model's parameters are computed (line 3). As presented in formula 3, these gradients are computed based on the local data and the current model parameters.

$$\nabla L_i = \nabla \text{Loss}(M_{local}, X_i, Y_i) \quad (3)$$

Finally, the client sends this gradient to the central server for aggregation and updating the global model (line 4). We note that clients send only the gradients of their local model parameters to the central server, rather than sending the entire model or raw data. This approach is essential to maintaining data privacy and reducing communication overhead.

Regarding the server side, its process is described in Algorithm 2. The server plays a pivotal role in managing the aggregation and updates of the global model by utilizing contributions from the devices participating in the training. To begin, the server initializes the global model denoted as M_{global} (line 1). This model serves as the starting point for the collaborative training process. Then, the algorithm proceeds by iterating through a predetermined number of rounds, indicated by the variable `nbrRound` (line 2). Within each round, the server performs several tasks. It first distributes the current global model M_{global} to the participating clients (line 3). Then, it sets up an accumulator named `aggregated_gradients` to gather the gradients sent by devices (line 4). After that and for each device enlisted in the list of participating devices, the server receives the gradients (`gradients_local`) transmitted by the device. The server subsequently aggregates these local gradients by adding them to the `aggregated_gradients` accumulator (lines 4-8). Once all the gradients have been aggregated, the server calculates the averaged gradients by dividing the accumulated `aggregated_gradients` by the total number of participating devices (line 9). Afterwards, the server employs the calculated averaged gradients to update the global model (M_{global}) through the `UpdateGlobalModel` function (line 10).

That means that the global model at iteration $t+1$, denoted as $M_{global_{t+1}}$, is obtained by aggregating the local models from the participating clients. In our architecture the aggregation is done using averaging method (where N is the number participating clients):

$$M_{global_{t+1}} = \frac{1}{N} \sum_{i=1}^N M_{global_t}^i \quad (4)$$

We note that $M_{global_{t+1}}$ is used as the new global model for the next iteration.

Algorithm 2 : Server Side

Input : *Devices*: the participating devices for training
nbrRound : positiveNumber
Output : an updated global model

```
1  $M_{global} \leftarrow InitializeGlobalModel()$ 
2 for (round  $\leftarrow 0$  to nbrRound - 1 ) do
3   broadcast  $M_{global}$  to all clients in Devices
4   aggregated_gradients  $\leftarrow 0$ 
5   for (each device in Devices) do
6     gradients_local  $\leftarrow ReceiveGradientsFromClient()$ 
7     aggregated_gradients  $\leftarrow aggregated\_gradients + gradients\_local$ 
8   end for
9   averaged_gradients  $\leftarrow aggregated\_gradients / nbr\_devices$ 
10   $M_{global} \leftarrow UpdateGlobalModel(M_{global}, averaged\_gradients)$ 
11 end for
12 return ( $M_{global}$ )
```

5. Some Implementation Aspects of our Architecture

5.1. Data Preparation

Data collection and preparation are fundamental steps that influence the entire ML pipeline, and in FML, they become even more critical due to the distributed and privacy-sensitive nature of the data. Properly preparing and managing data across distributed clients sets the foundation for successful federated learning. In this experiment, we used the same dataset as in our previous architecture. This allows us to compare the two obtained results. The data is collected from «Kaggle» known as predictive maintenance which offers a synthetic data set [17]. It reflects the actual predictive maintenance data of a system encountered in the industry. The dataset consists of 10,000 lines. Tab.1 presents the different columns of this dataset.

Table 1

Data set structure

<i>Column</i>	<i>Data type</i>
UDI	Int64
Product ID	Object
Type	Object
Air temperature [K]	Float64
Process temperature [K]	Float64
Rotational speed [rpm]	Int64
Torque [Nm]	Float64
Tool wear [min]	Int64
Target	Int64
Failure Type	Object

In our approach the most important stages of data preparation is density estimation. It refers to the process of estimating the underlying probability distribution of data within each client's local dataset without aggregating the actual data itself. This estimation is performed locally on individual clients. The results are used to guide the training process. In this step, we conducted initial experimentation to analyze and study the data from our clients. With the aim of evaluating our proposal, we partitioned the dataset across three distinct clients. This partitioning was carried out over multiple iterative phases. Indeed, the goal is to achieve a reasonable correlation among the datasets of the various clients.

Figure 3 shows a clear correlation among the data of the various clients, particularly in relation to the variables Rotational speed, Torque, and Tool wear. A significant correlation in

density estimation suggests that the data distributions across clients are aligned. This can simplify model training, aggregation, and result in faster convergence.

Figure 3 shows a clear correlation among the data of the various clients, particularly in relation to the variables Rotational speed, Torque, and Tool wear. A significant correlation in density estimation suggests that the data distributions across clients are aligned. This can simplify model training, aggregation, and result in faster convergence.

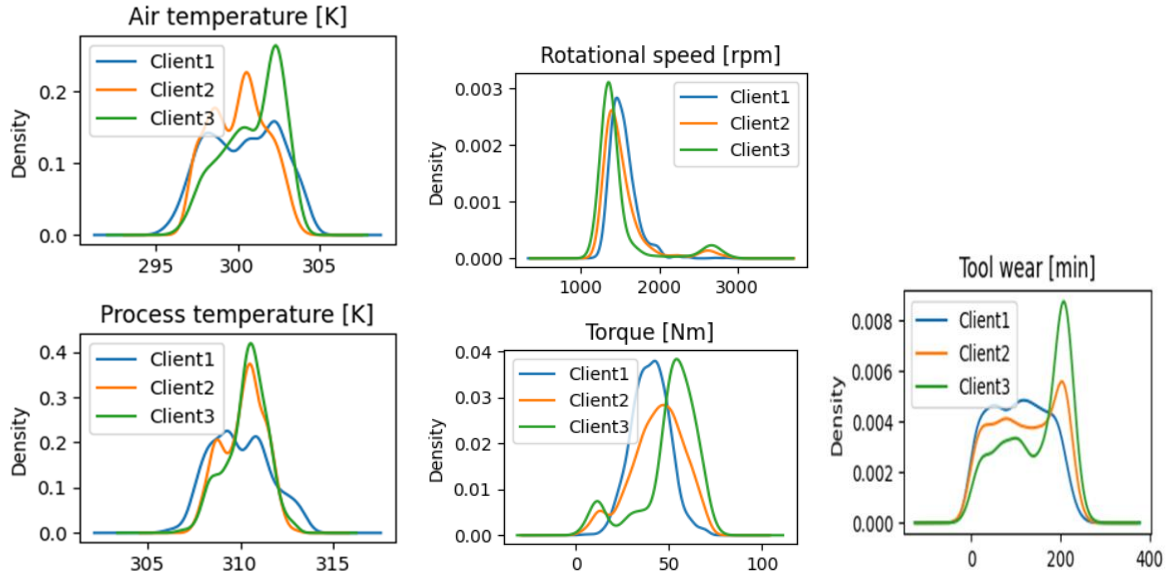


Figure 3. Data visualization

5.2. Experimental Setup and Training Process

In this section, we explain the experimental setup we used to implement our FML-based architecture. The TensorFlow Federated (TFF) [18] framework is employed for parallel client training, and the training process involves data batch division and distribution to individual clients. The federated model is constructed based on the same architecture as the global model and is trained using the Federated Averaging (FedAvg) algorithm [19].

As we mentioned previously, the training process centers on a collaborative training task involving both clients and a central server. The goal is to update a global model by utilizing the local models trained on each client's local data. The first step of the training process is the initialization of the global model. This step sets the model architecture and parameter values before engaging in collaborative training. It involves setting up the architecture of the learning model (specifying the number and types of layers, activation functions, and other architectural components). In our case, we define the architecture of the global model using TensorFlow's Keras API within TFF's context. We use the *tff.learning_from_keras_model* function to convert the Keras model into a TFF-compatible federated learning model.

For clients selection, we have used the TFF's built-in federated computations and functions. The *tff.federated* module provides functionalities for creating federated computations, including the selection of clients for each round of federated training. In our architecture, the *select_clients* function is defined as a federated computation using the *@tff.federated_computation* decorator. The function takes *client_weights* as an argument, which could be a federated integer type representing the weights of each client.

Regarding the step of local training (client update) we define the function local training as follow: *local_training(global_model, data_batch)*. The function takes as parameters the *global_model* as the model architecture and a *data_batch* representing the local data on the client

side. This function allows to initialize a copy from the global model using ***tff.learning.ModelWeights.from_model***. Local training is performed using TFF's gradient tape mechanism. The loss is computed based on the model's predictions and actual labels, and gradients are calculated. After executing the federated computation, the updated local model will contain the local model with updated parameters after the local training process. The resulting local model updates are sent to the central server.

For model aggregation we have used the Federated Averaging (FedAvg) algorithm. It permits aggregation local model updates from clients to update the global model. In our case, we define the function ***models_aggregation*** as a federated computation using the ***@tff.federated_computation*** decorator as follow: ***models_aggregation(global_model, local_models)***. The function takes as an input the global model and a list of local models representing the updated models from selected clients. In this function, the predefined ***with_weights*** method is used to attach the client weights to each local model. In addition, the predefined ***tff.federated_mean*** method is applied to compute the weighted average of the local models. This function calculates the weighted sum of model weights and divides it by the total weight to obtain the aggregated weights.

After the global model has been updated through the aggregation process, it needs to be broadcasted to all participating clients. This synchronization guarantees that each client has access to the most up-to-date global model for the subsequent round of training.

5.3. Discussion

By comparing federated learning and centralized learning based on the provided metrics (see Tab. 2), we observe the following results: federated learning achieves an accuracy of 0.95, a precision of 0.94, and a recall of 0.95. In contrast, centralized learning demonstrates higher values, with an accuracy of 0.97, a precision of 0.96, and a recall of 0.95.

Table 2

Federated vs. Centralized ML

	Recall	Accuracy	Precision
Federated	0.9584	0.9544	0.9420
Centralized	0.9580	0.9785	0.9695

Regarding accuracy, centralized learning outperforms federated learning, indicating a better ability to make overall correct predictions on the data. In terms of precision, both approaches maintain high precision values, with centralized learning slightly surpassing federated learning. This suggests that centralized learning is slightly better at avoiding false positive predictions. However, both centralized learning and federated learning exhibit the same recall value of 0.95, indicating equal performance in correctly identifying true positives.

The trade-offs between these approaches stem from Federated Learning's training on decentralized data from multiple clients, potentially leading to a more challenging optimization problem. On the other hand, Centralized Learning benefits from a centralized, potentially more comprehensive and well-curated dataset, which may account for its higher accuracy and precision.

Centralized learning is favoured when access to a large, well-curated, and representative dataset is available. This approach is particularly advantageous for tasks where centralizing data is feasible and practical. However, it's essential to acknowledge that in many real-world scenarios, especially in CPS, centralizing data may not always be a viable or desirable option CPS systems often involve distributed and interconnected devices or sensors that generate data in real-time.

6. Conclusion

In this paper, we have explored the development of adaptive CPS in dynamic environments, emphasizing the need for dynamic reconfiguration, resource optimization, and intelligent responses to ensure continued service quality. While ML techniques have been pivotal in addressing these challenges, they have often raised concerns regarding data security and privacy when centralized data processing is involved.

Our proposed approach introduced an innovative approach that leveraged FML technology to reconcile adaptability with data security and privacy. By adopting a federated approach, we demonstrated that CPS could become self-adaptive without compromising the confidentiality and privacy of sensitive data. This architecture offered a promising pathway for the development of self-adaptive and secure CPS.

As future work, we envision several directions for research and development:

- A crucial avenue is to conduct a comprehensive comparison of our FML-based architecture with other federated learning techniques, including FedProx and FedDP. This analysis will provide valuable insights into the relative strengths and weaknesses of each technique, particularly in the context of enhancing CPS adaptability.
- We intend to conduct experiments with a larger set of clients, explore varying data distributions, and assess the scalability of our architecture. These efforts will contribute to a more thorough understanding of its robustness and suitability for real-world, large-scale CPS deployments.
- We will focus also on the transition from controlled experimental settings to real-world CPS deployments. By assessing the performance and adaptability of our architecture in practical scenarios, we will validate its efficacy and its potential to address the evolving challenges of dynamic environments.

References

1. Hamzah, M., et al., *Distributed Control of Cyber Physical System on Various Domains: A Critical Review*. Systems, 2023. **11**(4): p. 208.
2. Lee, E.A., et al, *Cyber Physical Systems: Design Challenges*. In Proceedings of the IEEE, 2015.
3. Lesch, V., et al., *A literature review of IoT and CPS—What they are, and what they are not*. Journal of Systems and Software, 2023. **200**: p. 111631.
4. Gerostathopoulos, I., et al., *Tuning self-adaptation in cyber-physical systems through architectural homeostasis*. Journal of Systems and Software, 2019. **148**: p. 37-55.
5. Muccini, H., M. Sharaf, and D. Weyns. *Self-adaptation for cyber-physical systems: a systematic literature review*. in *Proceedings of the 11th international symposium on software engineering for adaptive and self-managing systems*. 2016.
6. Awad, M. and R. Khanna, *Efficient learning machines: theories, concepts, and applications for engineers and system designers* 2015: Springer nature.
7. Carvalho, T.P., et al., *A systematic literature review of machine learning methods applied to predictive maintenance*. Computers & Industrial Engineering, 2019. **137**: p. 106024.
8. Merabet, F.Z. and D. Benmerzoug, *Qos prediction for service selection and recommendation with a deep latent features autoencoder*. Computer Science and Information Systems, 2022. **19**(2): p. 709-733.
9. Wong, K.-S., et al., *An Empirical Study of Federated Learning on IoT-Edge Devices: Resource Allocation and Heterogeneity*. arXiv preprint arXiv:2305.19831, 2023.
10. Alzahrani, A., et al. *Improved Wireless Medical Cyber-Physical System (IWMCPs) Based on Machine Learning*. in *Healthcare*. 2023. MDPI.
11. Diaz-Rozo, J., C. Bielza, and P. Larrañaga, *Machine learning-based CPS for clustering high throughput machining cycle conditions*. Procedia Manufacturing, 2017. **10**: p. 997-1008.
12. Wu, M., Z. Song, and Y.B. Moon, *Detecting cyber-physical attacks in CyberManufacturing systems with machine learning methods*. Journal of intelligent manufacturing, 2019. **30**: p. 1111-1123.

13. Babouche, S., S. Ouchani, and M. Zghal. *An Adaptive Attack Prediction Framework in Cyber-Physical Systems*. in *2022 Ninth International Conference on Software Defined Systems (SDS)*. 2022. IEEE.
14. Graß, A., C. Beecks, and J.A.C. Soto. *Unsupervised anomaly detection in production lines*. in *Machine Learning for Cyber Physical Systems: Selected papers from the International Conference ML4CPS 2018*. 2019. Springer.
15. Chiu, M.-C., C.-D. Tsai, and T.-L. Li, *An integrative machine learning method to improve fault detection and productivity performance in a cyber-physical system*. *Journal of Computing and Information Science in Engineering*, 2020. **20**(2): p. 021009.
16. Azari, N., et al. *Fault Prediction Using Supervised and Unsupervised Learning Algorithms in Cyber Physical Systems*. in *2022 2nd International Conference on New Technologies of Information and Communication (NTIC)*. 2022. IEEE.
17. Matzka, S. *Explainable artificial intelligence for predictive maintenance applications*. in *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*. 2020. IEEE.
18. TFF. *TensorFlow Federated: Machine Learning on Decentralized Data*. 2023; Available from: <https://www.tensorflow.org/federated>.
19. McMahan, B., et al. *Communication-efficient learning of deep networks from decentralized data*. in *Artificial intelligence and statistics*. 2017. PMLR.