Efficient Vanilla Split Learning for Privacy-Preserving Collaboration in Resource-Constrained Cyber-Physical Systems

Nabila Azeri[1], Ouided Hioual[2] and Ouassila Hioual[2,3]
[1]ICOSI Laboratory, Abbes Laghrour University, 40004, Khenchela, Algeria
[2]Mathematics and Informatics Department, Abbes Laghrour University, 40004, Khenchela, Algeria
[3]LIRE Laboratory, Constantine 2 University, BP67A, Ali Mendjeli, 25000, Constantine, Algeria
E-mail: azeri.nabila@univ-khenchela.dz, hioual.ouided@univ-khenchela.dz, hioual_ouassila@univ-khenchela.dz

Keywords: cyber-physical systems, vanilla split machine learning, privacy-preserving learning, resource-constrained learning

Received: May 13, 2024

In the realm of Cyber-Physical Systems (CPS), the integration of Federated Machine Learning (FML) algorithms has become indispensable for enhancing adaptability, data privacy, and security. However, FML falls short when deployed in resource-constrained CPS environments due to its inherent demands on client resources. To address this limitation, this paper introduces a novel Split Machine Learning (SML) architecture tailored specifically for resource-constrained CPS deployments. Unlike FML, SML strategically splits the model between devices and a central server, enabling collaborative learning while preserving data privacy. Adopting a distributed learning paradigm, SML facilitates real-time system adaptation based on local sensor data, mitigating communication overhead and ensuring privacy. Experimental evaluation demonstrates that the proposed SML-based architecture achieves an accuracy rate of 97.56% with a processing time reduction of approximately 41% compared to FML methods. These results highlight the potential of our approach to improve collaboration in resource-constrained environments while maintaining high levels of privacy and performance.

Povzetek: Študija predstavlja arhitekturo razdeljenega učenja (SML) za kibernetsko-fizične sisteme z omejenimi viri, ki z razdelitvijo modela med naprave in strežnik zagotavlja visoko zasebnost, manjšo obremenitev ter dosega 97,56 % natančnost in 41 % hitrejšo obdelavo v primerjavi s federiranim učenjem.

1 Introduction

In the era of interconnected technologies, Cyber-Physical Systems (CPS) play a pivotal role in various applications, from smart infrastructures to autonomous systems [14]. However, the dynamic nature of the real world and evolving user requirements necessitate continuous adaptation for CPS to remain effective.

Machine Learning (ML) has emerged as a powerful tool for enhancing adaptability in CPS [18], [17], [16]. By analyzing sensor data and learning patterns, ML algorithms can guide system adjustments and optimize performance. However, conventional ML approaches often centralize data on a server, raising serious concerns regarding data privacy and security. This becomes particularly critical when dealing with sensitive data collected from personal devices or industrial processes.

To address these challenges, collaborative learning approaches such as Federated Machine Learning (FML) has been introduced [21], [4], [8]. This technique distributes the training process across devices, allowing models to learn from local data without directly sharing it. While FML offers advantages in data privacy, its limitations become evident in resource-constrained environments. Primarily, these limitations arise due to FML's significant demands on client resources. In such settings, where computational power and bandwidth are often limited, FML struggles to operate efficiently. The decentralized nature of FML requires participating devices to possess sufficient computational capabilities to handle model training tasks locally. However, in resource-constrained environments, such as those found in many CPS applications, devices may lack the necessary processing power or memory to execute these tasks effectively.

This paper introduces the concept of SML (Split Machine Learning) as a particularly advantageous approach, especially for resource-constrained CPS. SML further refines the collaborative learning concept by strategically splitting the model itself between devices and a central server. By doing so, it aims to address the limitations of centralized learning approaches and the resource demands of FML, thereby enhancing adaptability and data privacy in resource-constrained CPS environments. Specifically, this paper offers the following contributions::

– Integration of SML into our previous work on self-adaptive CPS architectures, enhancing adaptability while preserving data privacy and security, particularly in resource-constrained CPS environments.

– Application of our approach to a real-world case study in fault prediction within an industrial system, demonstrating its effectiveness in practical CPS deployments.

– Evaluation of the proposed SML-based approach through experimental analysis reveals promising outcomes, particularly in resource-constrained environments, with notable achievements in terms of accuracy and performance.

The rest of the paper is structured as follows. Section 2 discusses related work in the field. Section 3 presents an overview of our previous research efforts, particularly focusing on our earlier investigations and architectural proposals. Section 4 elaborates on the intricacies of the proposed SML-based architecture, elucidating its key components and operational dynamics. The practical application of our architecture on a real-world industrial system are detailed in Section 5. Section 6 offers concluding remarks, summarizing the findings and outlining some future work.

2 Related work

ML techniques integrated into CPS have been instrumental in addressing challenges like self-adaptation, security, and data privacy, enabling CPS to autonomously adapt to dynamic environments, enhance security measures, and safeguard sensitive data [9, 13]. This section surveys existing research on utilizing ML within CPS environments, highlighting its benefits in facilitating self-adaptation, strengthening security mechanisms, and ensuring data privacy, as demonstrated in Table 1. The table categorizes these works based on several key criteria, including the specific ML method employed, the learning process (centralized vs. distributed), and considerations for privacy, security, communication overhead, and resource constraints in CPS deployments. Following this overview, we transition to a subsection "Discussion," where we critically analyze the limitations of existing approaches and set the stage for introducing our novel contribution.

In [4], authors propose an innovative approach for enhancing self-adaptive CPS using FML. While many self-adaptation techniques in CPS emphasize performance gains, this work prioritizes security, data privacy, and adaptability. The proposed approach leverages FML technology to reconcile adaptability with data security and privacy, addressing the challenges posed by dynamic environments, shifting user requirements, and device dynamics.

Authors in [10] propose DeepFed, a federated deep learning scheme for detecting cyber threats in industrial CPS. The scheme utilizes a convolutional neural network (CNN) and a gated recurrent unit (GRU) to develop an intrusion detection model tailored for industrial CPS. Furthermore, a federated learning framework is introduced, allowing multiple industrial CPS to collaboratively build an intrusion detection model while preserving data privacy. A Paillier cryptosystem-based secure communication protocol is also devised to ensure the security and privacy of model parameters during the training process. Experimental results on real industrial CPS datasets demonstrate the effectiveness of DeepFed in detecting various cyber threats and its superiority over existing schemes.

In [1], authors discuss the integration of ML techniques within CPS, focusing on Industry 4.0. CPS combines computation and physical processes and ML optimizes CPS functionalities such as domain adaptation, system fine-tuning, and vulnerability detection. ML enables CPS to learn from large-scale data, enhancing security and privacy in industrial settings. The paper highlights ML applications in predictive maintenance, quality assurance, and optimization of manufacturing processes and supply chains, emphasizing ML's transformative impact on CPS in Industry 4.0.

In their paper, Wickramasinghe et al. [20] propose a methodology for explainable unsupervised ML tailored for CPS. They address the challenge of the 'black-box' nature of complex ML models by introducing explainable unsupervised ML models, particularly suitable for safety-critical CPS applications. Their approach utilizes Self-Organizing Maps (SOMs) based clustering methodology to generate both global and local explanations, enhancing the interpretability of ML models within CPS contexts. Through feature perturbation techniques, they evaluate the fidelity of the generated explanations, demonstrating the effectiveness of their proposed method in identifying the most important features responsible for decision-making processes in CPS.

In [12], M. Rouzbahani et al. delve into anomaly detection in CPS using ML methods. The authors underscore the complexity of CPS, which integrate cyber components into the physical world, leading to diverse tasks and close interactions. With the proliferation of smart features and communication tools in CPS, new challenges related to security and privacy have arisen, particularly in systems like the smart grid. Anomaly detection emerges as a crucial strategy for enhancing CPS security, yet comparing various detection methods poses challenges due to their diversity. To address this, the chapter focuses on ML-based anomaly detection methods, highlighting their effectiveness through a case study on False Data Injection (FDI) attacks. Through their exploration, the authors contribute insights into the application of ML techniques for anomaly detection in CPS, emphasizing their potential in mitigating security threats.

In [15], authors propose a novel approach for detecting faults in vehicular cyber-physical systems (VCPSs). They highlight the potential of VCPSs to enhance transportation safety, mobility, and sustainability through wireless communication. However, they also address the vulnerability of cloud-oriented architectures to cyber attacks, emphasizing the risks to safety, privacy, and property. Their proposed solution involves a neural network-based technique aimed at identifying and tracking fault data injection attacks in real time within a platoon of connected vehicles. They develop a decision support system to mitigate the probabil-

Table 1: Comparative overview of ML applications in CPS

Papers	**Goal**	**Method ML**	**Learning Process**	**Privacy Considerations**	**Security Considerations**	**Commu. Overhead**	**Resource-Constrained Considerations**
Azeri et al. [4]	Enhancing Self-Adaptive CPS using FML	Federated Learning	Distributed	Yes	Yes	Low	No
Li et al. [10]	FML for Intrusion Detection in Industrial CPS	Deep Learning (CNN and GRU)	Distributed	Yes	Yes	Low	No
Ahmed et al. [1]	Machine learning in CPS in Industry 4.0	Various ML techniques	Centralized	Yes	Yes	High	No
Wickram et al. [20]	Explainable unsupervised ML for CPS	Self-Organizing Maps	Centralized	No	No	High	No
Rouzbahani et al. [12]	Anomaly detection in CPS using ML	Various ML techniques	Centralized	No	Yes	High	No
Sargolzaei et al. [15]	Fault detection in VCPSs using neural network-based technique	Neural Network	Centralized	Yes	Yes	High	No
Alshboul et al. [2]	Predictive maintenance in concrete manufacturing using ML	Feature selection	Centralized	No	Yes	High	No
Zhang et al. [23]	Federated Learning-based Edge Computing Platform for CPS	Federated Learning	Distributed	Yes	Yes	Low	No
Xu et al. [22]	Multiagent federated reinforcement learning for secure incentive mechanism in CPS	Federated Learning	Distributed	Yes	Yes	Low	No
Guo et al. [7]	Deep federated learning for secure POI microservices in CPS	Deep federated learning	Distributed	Yes	Yes	Low	No
Our approach	**Collaboration Learning in resource-constrained CPS**	**Split Learning**	**Distributed**	**Yes**	**Yes**	**Low**	**Yes**

ity and severity of potential accidents resulting from these attacks, ultimately enhancing system reliability, robustness, and safety.

In [2], authors propose an empirical exploration of predictive maintenance in concrete manufacturing. By harnessing machine learning techniques, they aim to enhance equipment reliability in construction project management. The study identifies key features, such as 24-hour mean voltage, crucial for predicting machinery failure within the concrete manufacturing framework. Insights gleaned from this empirical investigation underscore the significance of integrating machine learning methodologies into construction project management practices. This integration not only improves the accuracy of maintenance forecasts but also reinforces equipment dependability, ensuring optimal efficacy and benefit in the concrete manufacturing paradigm.

Authors in [23] propose a novel FML-based Edge Computing platform named ”FengHuoLun” specifically designed for CPS. This platform aims to address the challenge of ensuring trustworthy smart services in Edge Computing environments by leveraging Federated Learning. With FengHuoLun, smart services can be implemented with machine learning models trained in a trusted FML framework, ensuring the trustworthiness of CPS behaviors through testing and monitoring.

In [22], authors introduce a novel approach employing multiagent federated reinforcement learning to devise a secure incentive mechanism in intelligent CPS. While federated learning addresses data privacy concerns in CPS, ensuring efficient incentive mechanisms remains crucial. Deep reinforcement learning is explored as a solution for long-term incentivization amidst dynamic environments. However, the heterogeneity of CPS devices poses a challenge, affecting the convergence rate of existing single-agent reinforcement learning. The proposed multiagent learning-based mechanism addresses this issue by approximating stationarity in federated learning with heterogeneous CPS. The approach formulates the secure communication and data resource allocation problem as a Stackelberg game and models it as a partially observable Markov decision process to handle device heterogeneity. A multiagent federated reinforcement learning algorithm is devised to efficiently learn allocation policies, mitigating policy evaluation variances caused by device interactions without compromising privacy.

In [7], the authors propose a deep federated learning framework to enhance secure points of interest (POI) microservices in CPS. This framework aims to improve data security by isolating the cloud center from accessing user data on edge nodes. Through interactive training between the cloud center and edge nodes, reliable deep-learning-based models are pre-trained on edge nodes, and parameter updating is coordinated via federated learning. The proposed approach is evaluated using real-world POI-related datasets, demonstrating optimal scheduling performance and practical utility.

2.1 Discussion

As shown in Table 1, numerous studies have made significant contributions towards improving security considerations (e.g., intrusion detection) and enhancing resilience (e.g., self-adaptation) in CPS environments. However, despite their contributions, limitations arise when considering real-world deployment, particularly regarding security, privacy, and resource constraints.

Limitations of Existing Approaches:

- Centralized Machine Learning: While effective in controlled settings, centralized ML architectures concentrate sensitive data in a central repository, making them susceptible to security breaches such as single point of failure vulnerabilities or insider threats. Additionally, the continuous transmission of data to a central server for training and updates incurs substantial communication overhead, hindering scalability in large-scale CPS deployments.
- Federated Machine Learning: Although FML offers advantages in data privacy and communication efficiency by keeping training data on local devices, its decentralized nature presents challenges in resource-constrained environments. Limited processing power, memory, and battery life on sensor devices can hinder effective local model training, hindering widespread adoption in CPS settings where efficient resource utilization is paramount.

Advantages of our approach:

Our SML-based approach serves as a promising solution that addresses the limitations of both centralized ML and FML. By partitioning the training process between client devices and a central server, our approach offers several advantages:

- Enhanced Data Privacy: Sensitive data remains largely on local devices, minimizing the risk of exposure through breaches in centralized storage.
- Reduced Communication Overhead: Only a portion of the training data needs transmission to the server, significantly reducing communication costs compared to centralized approaches.
- Efficient Resource Utilization: Local devices perform computations on smaller datasets, alleviating the burden on resource-constrained CPS nodes. This makes SML well-suited for large-scale deployments in CPS environments.

Before introducing the novel contribution of this paper, we briefly present our previous work on the multilayer architecture for adaptive CPS in the next section. Understanding these prior developments is crucial for contextualizing our current approach, which builds upon this foundation.

3 Our previous works

3.1 Proposal of an architecture for CPS

In our previous architecture for self-adaptive CPS [3], we designed a comprehensive framework comprising essential software modules to ensure CPS functionalities. These modules encompassed resource and data management, process planning, and the transformation of machines into self-aware, self-learning, and self-reconfiguring entities. Illustrated in Fig. 1, our architecture consisted of distinct layers:

Physical layer: Serving as the foundation, this layer encompassed the local assembly of machines connected to the Internet via the OPC-UA protocol, facilitating standardized communication between different units. It primarily featured sensors responsible for collecting signals from machines and actuators for translating electrical signals into physical movement.

Data/Resource Processing Layer: This layer facilitated the collection of data from diverse sources, directly interacting with data producers at the physical layer and data storage in the Edge/Cloud. Modules within this layer included the conditioner, data management, resource management, planning process, monitoring and control, and request management.

Data Storage Layer: Serving as the storage backbone for the CPS, this layer was distributed between the Edge and the Cloud, depending on the nature of the data and the processing time required.

Learning Application Layer: At the forefront of adaptive functionalities, this layer facilitated resource allocation, QoS analysis, process optimization, predictive maintenance, and fault detection. Here, various ML techniques such as regression, classification, clustering, and reinforcement learning were employed. These models were trained meticulously on a comprehensive dataset, combining historical knowledge with real-time information to make predictions and informed decisions, driving the system's adaptive capabilities.

3.2 Integration of FML into our architecture

Our centralized learning based architecture exhibited commendable accuracy and precision. However, it encountered constraints related to data security, privacy issues, and communication overhead. These limitations highlighted the need for an alternative strategy that could retain the advantages of precise predictions while addressing the drawbacks associated with centralized data handling and computation.

Our work in [4] addressed these limitations by incorporating FML into the architecture. This FML-based approach distributes processing and intelligence to the network's edge, enabling local data analysis and decision-making. By using FML, the architecture facilitates collaboration among edge devices to train ML models without requiring centralized data collection. This decentralized approach offers several advantages: it preserves data privacy, reduces communication overhead, and enhances system resilience, all while maintaining the high accuracy and precision achieved by the initial centralized architecture.

3.3 Practical implementation and performance evaluation

To validate the efficacy and real-world applicability of both versions of our architecture, we implemented them in a practical context as described in [4, 5]. This application aligns with the realm of failure prediction using supervised and unsupervised learning algorithms. 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.

In [5], we detailed the different ML algorithms used for implementing the principle of fault prediction in CPS. Experiments demonstrated the effectiveness of our approach in fault prediction, with achieved accuracy surpassing 95%.

In [4], we detailed the training process in our FML-based architecture. In this experiment, we utilized the same dataset as in our previous work, enabling a direct comparison of the obtained results. This dataset will also be utilized in this paper for comparison purposes.

In this paper, our goal is to significantly improve upon our previous work in the context of resource-constrained CPS by adopting the SML paradigm. The shift to SML architecture is motivated by the need to address challenges related to data privacy, security, and resource constraints, which are particularly pertinent in CPS environments. Our objective is to maintain the accuracy and precision of the learning model while enabling more efficient utilization of resources and greater resilience in dynamic operating conditions.

4 The proposed SML-based architecture

In our previous FML-based architecture, the collaborative training process is distributed between the server and clients. Although this approach ensures data privacy, it demands relatively higher computational resources on the client side, which might be a constraint in resource-constrained environments, such as CPS. The decentralization of training in FML, while preserving privacy, can pose challenges for devices with limited computational capabilities.

Recognizing the importance of deploying machine learning models in CPS scenarios, where resource constraints are prevalent, we transition to SML to address these challenges effectively. In SML, the training process is shifted predominantly to the server side, aligning with the characteristics of resource-constrained devices. This shift allows for more efficient utilization of available resources, making SML a suitable alternative for applications in CPS.

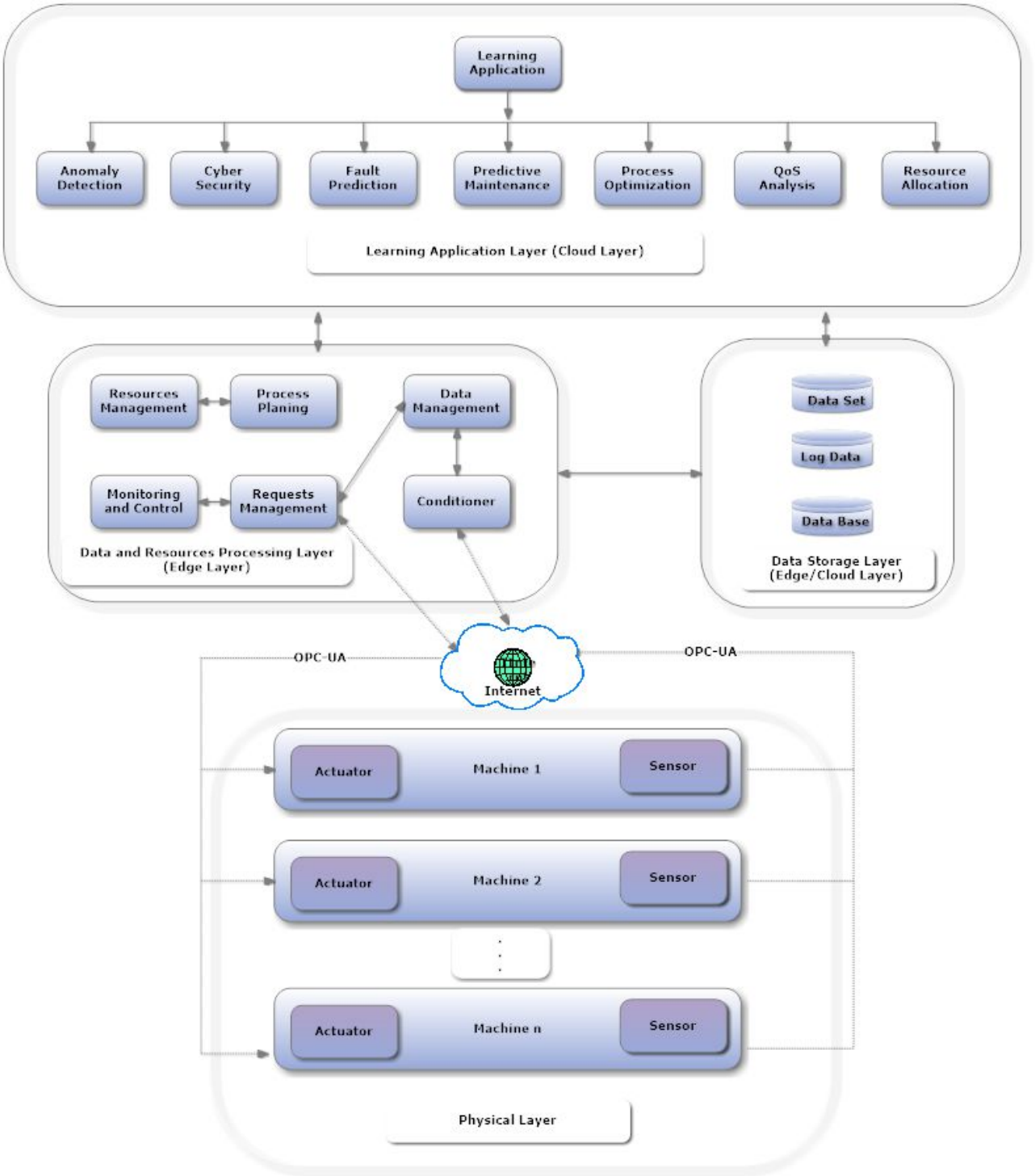

Figure 1: The proposed Multi-layer architecture for CPS

The following subsections delve into the Split Learning configuration used in our proposed SML-based approach, as well as its key components and operational dynamics, highlighting its advantages in addressing the challenges posed by resource-constrained environments.

4.1 Split learning configurations

SML encompasses various configurations tailored to specific needs and constraints. Three common configurations include [19]:

- Simple Vanilla Configuration for Split Learning: In this configuration, each client trains a partial deep network up to a designated layer, known as the cut layer. The outputs from this layer are forwarded to a central server for further model training without exposing raw data. The server aggregates gradients and sends them back to clients for local backpropagation. This setup minimizes data exposure while facilitating collaborative model training.
- U-shaped Configurations for Split Learning without Label Sharing: These configurations mitigate the need for label sharing among clients while preserving data privacy. The network architecture is wrapped around at the end layers of the server's network, and outputs are sent back to clients.
- Vertically Partitioned Data for Split Learning: This configuration enables collaborative model training across multiple institutions with different data modalities without sharing raw data. Each institution trains a partial model up to its designated cut layer, and outputs are concatenated and forwarded to a central server for further training.

For resource-constrained CPS, simplicity, efficiency, and privacy are critical considerations. The Vanilla configuration of split learning aligns with these requirements for the following reasons:

- Simplicity and Efficiency: Vanilla Split Learning prioritizes a simple implementation, minimizing both computational and communication overhead. In resource-constrained CPS environments, where computational resources and network bandwidth are limited, simplicity is essential for efficient model training and inference.
- Data Privacy Preservation: By keeping raw data local to each client and only sharing model updates with the central server, Vanilla split learning ensures robust data privacy. This decentralized approach minimizes the risk of data breaches or privacy violations, which is crucial in CPS applications handling sensitive information.
- Scalability and Adaptability: Vanilla split learning's decentralized nature makes it highly scalable and adaptable to diverse CPS deployments. Clients can operate independently, enabling seamless integration with edge devices. This scalability facilitates the expansion of CPS deployments without compromising performance or security.
- Flexibility in Model Customization: Vanilla split learning allows clients to customize their local model architectures and training data to suit specific CPS requirements. This flexibility enables adaptation to varying environmental conditions and application domains, enhancing the overall resilience and effectiveness of the CPS.

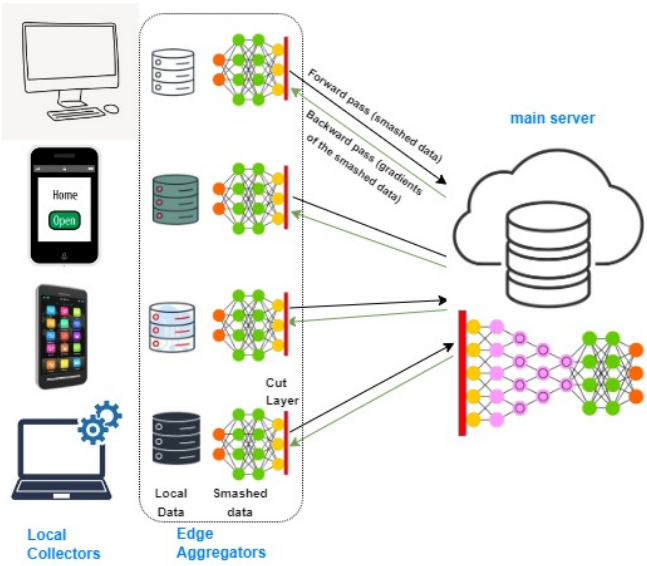

Figure 2: The proposed Vanilla SML-based architecture

4.2 Architecture components

In this subsection, we outline the key components of our Vanilla SML-based architecture and illustrate how they operate together. We focus on the enhancements introduced compared to our previous architecture presented in Figure 1. As illustrated in Figure 2, the architecture comprises three main entities: local data collectors, edge aggregators, and the main server.

Local Data Collectors:

Local Data Collectors play a crucial role in managing data within a group or federation. They gather information from various sources, including industrial equipment, smartphones, IoT devices, sensors, and other endpoints. These *Local Data Collectors* prioritize data security and privacy. Often, they achieve this by aggregating or summarizing the collected data before forwarding it to the next

processing stage, the Edge Aggregator. In essence, *Local Data Collectors* act as intermediaries, preparing and transmitting relevant information while ensuring adherence to privacy and security constraints.

Edge Aggregators:

In our architecture, the *Edge Aggregator* operates as the client-side component responsible for managing the client-side model (M_c). In this collaborative learning setup, each client, represented by an *Edge Aggregator*, possesses a unique local dataset (D_i). Notably, the *Edge Aggregator* also plays a role in collecting and managing data from local collectors, which include a diverse range of devices such as industrial equipment, smartphones, IoT devices, sensors, and other endpoints.

The client-side model (M_c) is initialized with random weights, and local training is conducted on the initial layers of the neural network, up to a designated cut layer that separates the client-side and server-side segments.

During training iterations, forward propagation generates ”smashed data” up to the cut layer, representing activations. This smashed data is securely transmitted to the central server for further processing through the remaining layers of the global model. The *Edge Aggregator* manages the client-side model (M_C), orchestrates the training process, and actively collects data from local collectors to contribute to the collaborative learning paradigm.

Main server:

The *Main server* functions as the central server in our Vanilla SML architecture, orchestrating the collaborative training process across multiple Edge Aggregators. In our case, the server communicates with clients in a sequential manner to ensure a structured learning process. The sequential communication involves interacting with each client in a sequence (Client 1, Client 2, ..., Client n) for both forward and backward passes (see figure 2). This approach aims to enhance the learning process by iteratively refining the global model through multiple interactions.

In our architecture, the communication between the *Main server* and Edge Aggregators is a key aspect of the learning process. During the forward pass, smashed data (activations from the split layer, also known as the cut layer) is transmitted from the client-side network to the server, allowing the global model to process the data through its remaining layers. The backward pass involves transmitting gradients generated at the server's first layer back to the Edge Aggregators, contributing to the refinement of the client-side model.

4.3 Collaborative training in our architecture

In our Vanilla SML-based architecture, the primary objective is collaborative model training, wherein the server collaborates with each client in sequence for model training. Each client possesses an individual local dataset denoted as D_i.

In our architecture, the training process comprises a Client (C) and a server (S). A global model, denoted as M_{global}, is created, which consists of two distinct parts: M_c (Client-side Model) and M_s (Server-side Model). Algorithm 1 outlines the behavior of each client within our architecture. The M_c is responsible for processing the initial layers of the neural network, conducting local training on the client's unique dataset, and generating ”smashed data” after forward propagation up to a designated cut layer. On the other hand, the M_s, residing on a central server, utilizes the smashed data received from the client to complete forward propagation on the remaining layers of the global model. The server side process is described in Algorithm 2. The server conducts forward propagation on the server-side model (M_s) using the received smashed data to obtain to minimize a loss function : $L_i(\text{smash\_data})$. This is done by the formula:

$$M_s = \text{argmin}_M L_i(\text{smash\_data}) \tag{1}$$

After the training phase, the gradients ∇L_i of the loss function L_i with respect to the model's parameters are calculated. These gradients are determined using the $smash\_data$ and the current model parameters, as shown in formula 2.

$$\nabla L_i = \nabla Loss(M_s, \text{smash\_data}) \tag{2}$$

This process involves computing the loss function, engaging in backpropagation, and updating its own weights until reaching the cut layer. The gradients corresponding to the smashed data are then communicated back to the client. This collaborative training process between the M_c and the M_s iterates until convergence, contributing to the refinement of the M_{global}.

5 Experimental evaluation

5.1 Data collection

The dataset, sourced from Kaggle and described in [11], represents a synthetic dataset reflecting real predictive maintenance scenarios. Figure 3 provides an overview of the dataset columns, showcasing its structure. The dataset consists of data points with the following main features:

- **Product ID:** Each product is assigned a unique identifier consisting of a letter denoting the product quality variant (L for low, M for medium, and H for high) and a variant-specific serial number.
- **Air Temperature [K]:** Generated through a random walk process and subsequently normalized to have a standard deviation of 2 K around the mean temperature of 300 K.

Algorithm 1 Client-Side Algorithm

Require: M_c : Client-side Model with random weights
D_c : Local data on the client side
num_epochs : Number of training epochs
Ensure: Smashed data for the server
for epoch $\leftarrow 1$ to num_epochs **do**
  $M_c \leftarrow TrainLocalModel(M_c, D_c)$
  $smash\_data \leftarrow M_c.get\_smash\_data()$
  $SendToServer(smash\_data)$
  $gradients\_from\_server \leftarrow ReceiveFromServer()$
  $M_c.backward(gradients\_from\_server)$
end for

Algorithm 2 Server-Side Algorithm

Require: M_s : Initial server-side model
$clients$: List of clients
Ensure: Updated gradients
for client in clients **do**
  for epoch $\leftarrow 1$ to num_epochs **do**
    $smash\_data \leftarrow receive\_from\_client(client)$
    $predictions \leftarrow M_s.forward(smash\_data)$
    $loss \leftarrow calcul\_loss(predictions, smash\_data)$
    $M_s.backward(loss)$
    $send\_to\_client(client, M_s.get\_gradients())$
  end for
end for

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   UDI                     10000 non-null  int64
 1   Product ID              10000 non-null  object
 2   Type                    10000 non-null  object
 3   Air temperature [K]     10000 non-null  float64
 4   Process temperature [K] 10000 non-null  float64
 5   Rotational speed [rpm]  10000 non-null  int64
 6   Torque [Nm]             10000 non-null  float64
 7   Tool wear [min]         10000 non-null  int64
 8   Target                  10000 non-null  int64
 9   Failure Type            10000 non-null  object
dtypes: float64(3), int64(4), object(3)
memory usage: 781.4+ KB
```

Figure 3: Dataset structure

- **Process Temperature [K]:** Produced using a random walk process, normalized to a standard deviation of 1 K, and added to the air temperature plus 10 K.
- **Rotational Speed [rpm]:** Derived from a power of 2860 W and superimposed with normally distributed noise.
- **Torque [Nm]:** Torque values follow a normal distribution around 40 Nm with a standard deviation of 10 Nm, ensuring no negative values.
- **Tool Wear [min]:** Tool wear duration varies based on product quality variants, with high, medium, and low variants adding 5, 3, and 2 minutes, respectively, to the total tool wear during the process.

Additionally, in this dataset, we have the 'Target' label that indicates whether the machine has failed in this particular datapoint. If the process fails, the 'Target' label is set to 1.

The choice of this dataset enables a meaningful comparison of results with our previous work. Throughout the data preparation phase, privacy considerations were paramount, aligning with the privacy-preserving principles of SML. To assess our proposal, we divided the dataset among three separate clients, ensuring a representative distribution of data and facilitating a comprehensive evaluation of our SML-based approach. In line with this, we utilized a synthetic dataset representing real predictive maintenance scenarios, divided among three clients. Each client's model

was trained locally using TensorFlow Keras, with a sequential server-client communication model.

The subsequent sections elaborate on the implementation of our Vanilla SML approach using TensorFlow Keras, providing insights into the architecture and training process. We then analyze the convergence of loss and accuracy during SML training, examining how these metrics evolve over time and evaluating the stability and efficacy of our approach. Following this, we present a comparative analysis, wherein we juxtapose the performance of our SML-based approach with existing methods, highlighting its advantages and contributions in the realm of resource-constrained CPS deployments.

5.2 Vanilla SML implementation with TensorFlow Keras

Central to the SML paradigm are the client models, which encapsulate domain-specific knowledge while training on local datasets. Utilizing the TensorFlow Keras API [6], we instantiate client models with architectures tailored to accommodate the dimensions and characteristics of local data. Each client model comprises multiple densely connected layers, incorporating activation functions such as ReLU to introduce non-linearity and facilitate model convergence. In our case, client models are created using the `create_client_model()` function, which defines the architecture and compiles the models.

Client models are trained iteratively across multiple epochs using the `fit()` function from the TensorFlow Keras API, with each epoch encompassing forward and backward propagation facilitated by the `train_on_batch()` method. This method updates model parameters using stochastic gradient descent. Training parameters such as batch size and learning rate are meticulously tuned using functions such as `compile()` and `fit()` to optimize convergence while mitigating computational overhead. In our case, the `compile()` function configures the model for training, specifying the Adam optimizer for efficient gradient descent and the binary cross-entropy loss function. This loss function quantifies the difference between model predictions and ground truth labels during binary classification tasks.

During training, client models exclusively access and learn from their respective local datasets, preserving data privacy and confidentiality. This is facilitated by functions such as `fit()`, which train the model on local data without sharing it externally. Following the client model construction, a central server model is also created, adhering to the principles of Vanilla SML. We utilize the `Sequential()` constructor from the TensorFlow Keras API to define the server model's architecture. This architecture is designed to be compatible with the client models, using the function `Dense()` to add densely connected layers. The server model acts as the central nexus for knowledge aggregation within the SML paradigm. During the training process, client models train locally and extract relevant information (e.g., gradients) after a designated cut layer. These are then sent to the server for aggregation. In our case, the server uses the functions `get_weights()` and `set_weights()` to retrieve these gradients and incorporate them into its own model, facilitating the collaborative learning process. This iterative exchange of gradients between clients and the server continues until a convergence criterion is met.

Following construction, the server model is compiled using the `compile()` function to prepare for training and knowledge aggregation, entailing specifying optimization algorithms, loss functions, and optional evaluation metrics. In our implementation, the Adam optimizer is employed for efficient gradient descent optimization, while binary cross-entropy serves as the loss function for binary classification tasks. Evaluation metrics, including accuracy, provide insights into model performance and convergence during training and aggregation phases, facilitated by functions such as `evaluate()`.

5.3 Convergence analysis of loss and accuracy during SML training

As elucidated earlier, the training process hinges on a synergistic collaboration between the server and the clients. This collaborative endeavor unfolds across multiple epochs, aiming to minimize the loss function and maximize the model's accuracy.

The central element of the training process is the computation of the loss value on the server side. This loss value serves as a crucial metric, quantifying the disparity between the model's predictions and the actual target values in the training dataset. A smaller loss value indicates a closer alignment between the model's predictions and the smashed data points, signifying an improved predictive capability.

This approach ensures that the training process not only refines the model's parameters but also steers it towards a state where its predictions better capture the underlying patterns within the training data.

Figure 4 depicts the relationship between the number of epochs and both the loss and accuracy for each client. It is evident that the number of training epochs plays a crucial role in shaping the loss values. During the initial training epochs, a substantial reduction in the loss is observed, indicating significant improvements as the model learns from client data. As training progresses (epochs = 70), the loss stabilizes at a minimum value, representing the optimal performance achievable with the given data and model architecture. Trying to add more epochs shows that the loss levels stabilize, indicating that the model has reached a point where further training doesn't result in substantial improvement, signifying convergence. Convergence signifies that the model has effectively captured patterns and relationships present in the training data.

Regarding accuracy, the graph illustrates dynamic changes during the training process. As the number of training epochs increases, accuracy steadily rises until it attains a stable value, reaching this stability at around 70 epochs.

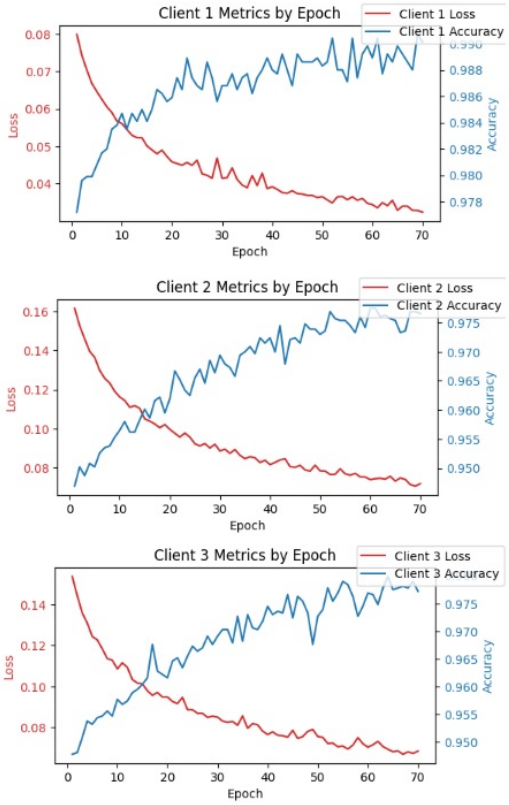

Figure 4: Loss and accuracy convergence for each client

This stability suggests that the model has learned comprehensively, performing optimally on the provided dataset. The attainment of a consistent accuracy level indicates that the model has successfully learned and adapted to the underlying patterns in the training data. Notably, at each epoch, the server leverages the smashed data received from the clients to enhance the global model, contributing to the continuous improvement in accuracy. After an initial increase during the early epochs, the accuracy values for all three clients stabilized at an average of 0.9756.

5.4 Comparative analysis

Table 2 presents the results obtained by the approach proposed in this paper alongside those obtained in our previous work [4, 5]. It's important to note that we utilized the same dataset for the sake of comparison. Like presented in Table 2, it is striking that both the centralized and SML techniques show similar degrees of accuracy, with both achieving accuracies up to 97%. This parity in accuracy underscores the viability of SML as a competitive alternative to centralized learning approaches.

To further assess the performance of decentralized approaches in resource-constrained environments, we delve into a comparative analysis between FML and SML approaches. While Table 2 provides insights into the accuracy metrics across different ML approaches, additional metrics are necessary to evaluate their efficiency in resource-

Table 2: Comparison of accuracy metrics across different ML approaches

ML approach	Accuracy
SML	0.9756
FML	0.9544
Centralized	0.9785

limited contexts. Specifically, we focus on the "Learning time for each client" metric, which sheds light on the computational resource utilization of each approach.

Maintaining consistent experimental configurations for FML and SML ensures a fair comparison. This includes using identical client configurations and dataset distributions across clients. Such uniformity enables an accurate evaluation and comparison of the performance of FML and SML under resource-constrained environments.

As presented in Figure 5, the results obtained from the experiments reveal notable differences in the learning time for each client between FML and SML. In resource-constrained environments, where efficient resource utilization is paramount, SML demonstrates a clear advantage. The learning time for each client in SML is significantly shorter compared to FML, indicating that clients utilizing SML consume fewer computational resources.

Specifically, the experimental results are summarized in Table 3, which shows the comparison of learning time between FML and SML for each client.

On average, across all clients, the SML approach achieves a reduction in learning time of approximately 41%. These results highlight the efficiency of SML in reducing learning time, which is critical for real-time applications in CPS. By significantly lowering the computational load on clients, SML not only enhances learning speed but also conserves resources, making it a highly suitable approach for resource-constrained environments.

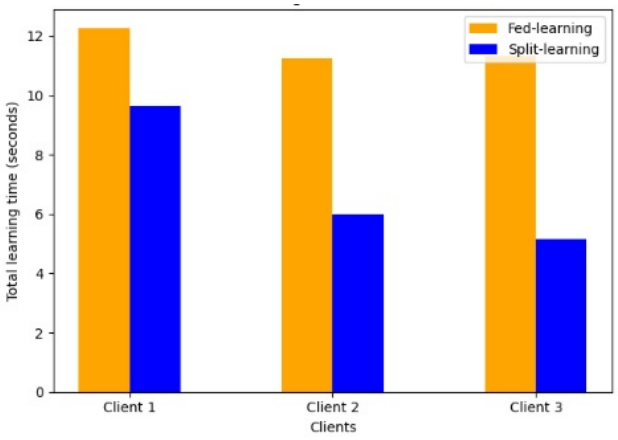

Figure 5: Total learning time for each client in FML and SML

Consequently, SML emerges as a more efficient and suit-

Client	Learning Time in FML	Learning Time in SML	Reduction (%)
Client 1	12.2 seconds	9.5 seconds	22.13%
Client 2	11.3 seconds	6.0 seconds	46.90%
Client 3	11.3 seconds	5.0 seconds	55.75%

Table 3: Comparison of learning time between FML and SML for each client

able approach for resource-constrained CPS deployments. Its ability to optimize computational resource utilization makes it a compelling alternative to FML in such environments. By effectively addressing the challenges associated with limited computational resources, SML showcases its potential to enhance the performance and adaptability of CPS systems operating under resource constraints.

However, it is important to acknowledge some potential limitations of the SML approach. Firstly, like in FML, SML relies on a central server for model aggregation, which could become a single point of failure. Implementing fault-tolerant mechanisms or exploring decentralized alternatives could mitigate this risk. Secondly, despite reducing communication overhead compared to centralized approaches, SML still requires transmitting model updates. Optimizing this further or managing it in networks with limited bandwidth remains an area for improvement. Furthermore, differences in local data distributions and training processes could lead to model inconsistencies, highlighting the importance of ensuring convergence and model stability across different clients. Moreover, the overall performance of SML can still be affected by network latency, especially in environments with poor connectivity. Strategies to minimize the impact of network delays should be explored.

6 Conclusion

This paper has introduced a novel SML-based architecture specifically designed to address the dual challenges of data privacy and resource constraints in CPS. Our proposed architecture utilizes the distributed learning power of SML to enable real-time system adaptation based on local sensor data, while simultaneously preserving data privacy.

Our experimental evaluation highlights the effectiveness of the proposed SML-based architecture. With an achieved accuracy of 97%, SML demonstrates competitive performance when compared to centralized learning approaches, surpassing the accuracy achieved by FML. Notably, the learning time for each client in SML is shorter than FML, making it a practical choice for resource-constrained CPS deployments.

Our future work will focus on further refining the SML architecture. We will explore different configurations, such as U-shaped or vertically partitioned approaches, to potentially improve efficiency and accuracy. Additionally, we aim to investigate fault-tolerant mechanisms and decentralized alternatives to reduce the risk of a single point of failure. Furthermore, we will optimize the communication overhead and manage model updates more efficiently, especially in networks with limited bandwidth.

Acknowledgements

The author would like to thank the anonymous reviewers for their valuable comments and suggestions, which were helpful in improving the paper.

References

[1] Rania Salih Ahmed, Elmustafa Sayed Ali Ahmed, and Rashid A Saeed. Machine learning in cyber-physical systems in industry 4.0. In *Artificial intelligence paradigms for smart cyber-physical systems*, pages 20–41. IGI global, 2021.

[2] Odey Alshboul, Rabia Emhamed Al Mamlook, Ali Shehadeh, and Tahir Munir. Empirical exploration of predictive maintenance in concrete manufacturing: Harnessing machine learning for enhanced equipment reliability in construction project management. *Computers & Industrial Engineering*, page 110046, 2024. doi:10.1016/j.cie.2024.110046.

[3] Nabila Azeri, Ouassila Hioual, and Ouided Hioual. Towards an approach for modeling and architecting of self-adaptive cyber-physical systems. In *2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–7. IEEE, 2022. doi:10.1109/pais56586.2022.9946921.

[4] Nabila Azeri, Ouided Hioual, and Ouassila Hioual. Enhancing self-adaptive cyber-physical systems using federated machine learning. In *TACC 2023 : 3rd Tunisian-Algerian Joint Conference on Applied Computing*, pages 108–119. ceur-ws.org, 2023.

[5] Nabila Azeri, Zeinb Zouikri, Meriem Rezgui, Ouided Hioual, and Ouassila Hioual. 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)*, pages 1–6. IEEE, 2022. doi:10.1109/ntic55069.2022.10100404.

[6] Multi framework deep learning API. Keras: The high-level api for tensorflow, 2023. https://www.tensorflow.org/guide/keras.

[7] Zhiwei Guo, Keping Yu, Zhihan Lv, Kim-Kwang Raymond Choo, Peng Shi, and Joel JPC

Rodrigues. Deep federated learning enhanced secure poi microservices for cyber-physical systems. *IEEE Wireless Communications*, 29(2):22–29, 2022. doi:10.1109/mwc.002.2100272.

[8] Xianting Huang, Jing Liu, Yingxu Lai, Beifeng Mao, and Hongshuo Lyu. Eefed: Personalized federated learning of execution&evaluation dual network for cps intrusion detection. *IEEE Transactions on Information Forensics and Security*, 18:41–56, 2022. doi:10.1109/tifs.2022.3214723.

[9] Sangjun Kim and Kyung-Joon Park. A survey on machine-learning based security design for cyber-physical systems. *Applied Sciences*, 11(12):5458, 2021. doi:10.3390/app11125458.

[10] Beibei Li, Yuhao Wu, Jiarui Song, Rongxing Lu, Tao Li, and Liang Zhao. Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Transactions on Industrial Informatics*, 17(8):5615–5624, 2020. doi:10.1109/tii.2020.3023430.

[11] Stephan Matzka. Explainable artificial intelligence for predictive maintenance applications. In *2020 third international conference on artificial intelligence for industries (ai4i)*, pages 69–74. IEEE, 2020. doi:10.1109/ai4i49448.2020.00023.

[12] Hossein Mohammadi Rouzbahani, Hadis Karimipour, Abolfazl Rahimnejad, Ali Dehghantanha, and Gautam Srivastava. Anomaly detection in cyber-physical systems using machine learning. *Handbook of big data privacy*, pages 219–235, 2020. doi:10.1007/978-3-030-38557-6_10.

[13] Felix O Olowononi, Danda B Rawat, and Chunmei Liu. Resilient machine learning for networked cyber physical systems: A survey for machine learning security to securing machine learning for cps. *IEEE Communications Surveys & Tutorials*, 23(1):524–552, 2020. doi:10.1109/comst.2020.3036778.

[14] Mutaz Ryalat, Hisham ElMoaqet, and Marwa AlFaouri. Design of a smart factory based on cyber-physical systems and internet of things towards industry 4.0. *Applied Sciences*, 13(4):2156, 2023. doi:10.3390/app13042156.

[15] Arman Sargolzaei, Carl D Crane, Alireza Abbaspour, and Shirin Noei. A machine learning approach for fault detection in vehicular cyber-physical systems. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 636–640. IEEE, 2016. doi:10.1109/icmla.2016.0112.

[16] Zakir Ahmad Sheikh, Yashwant Singh, Pradeep Kumar Singh, and Kayhan Zrar Ghafoor. Intelligent and secure framework for critical infrastructure (cps): Current trends, challenges, and future scope. *Computer Communications*, 193:302–331, 2022. doi:10.1016/j.comcom.2022.07.007.

[17] Rama Mercy Sam Sigamani. Adoption of machine learning with adaptive approach for securing cps. In *Handbook of Research on Machine and Deep Learning Applications for Cyber Security*, pages 388–415. IGI Global, 2020. doi:10.4018/978-1-6684-6291-1.ch061.

[18] Theocharis Theocharides, Muhammad Shafique, Jungwook Choi, and Onur Mutlu. Guest editorial: Robust resource-constrained systems for machine learning. *IEEE Design & Test*, 37(2):5–7, 2020. doi:10.1109/mdat.2020.2971201.

[19] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[20] Chathurika S Wickramasinghe, Kasun Amarasinghe, Daniel L Marino, Craig Rieger, and Milos Manic. Explainable unsupervised machine learning for cyber-physical systems. *IEEE Access*, 9:131824–131843, 2021. doi:10.1109/access.2021.3112397.

[21] Kok-Seng Wong, Manh Nguyen-Duc, Khiem Le-Huy, Long Ho-Tuan, Cuong Do-Danh, and Danh Le-Phuoc. An empirical study of federated learning on iot-edge devices: Resource allocation and heterogeneity. *arXiv preprint arXiv:2305.19831*, 2023.

[22] Minrui Xu, Jialiang Peng, Brij B Gupta, Jiawen Kang, Zehui Xiong, Zhenni Li, and Ahmed A Abd El-Latif. Multiagent federated reinforcement learning for secure incentive mechanism in intelligent cyber–physical systems. *IEEE Internet of Things Journal*, 9(22):22095–22108, 2021. doi:10.1109/jiot.2021.3081626.

[23] Chong Zhang, Xiao Liu, Xi Zheng, Rui Li, and Huai Liu. Fenghuolun: A federated learning based edge computing platform for cyber-physical systems. In *2020 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops)*, pages 1–4. IEEE, 2020. doi:10.1109/percomworkshops48775.2020.9156259.