Enhancing the Aggregation of the Federated Learning for the Industrial Cyber Physical Systems

Souhila Badra Guendouzi, Samir Ouchani, Mimoune Malki

Enhancing the Aggregation of the Federated Learning for the Industrial Cyber Physical Systems

Souhila Badra GUENDOUZI
LabRi-SBA laboratory,
Ecole supérieure en Informatique,
Sidi Bel-abbes, Algeria
LINEACT CESI, Lyon, France
b.guendouzi@esi-sba.dz

Samir OUCHANI
LINEACT CESI
Aix-en-Provence, France
souchani@cesi.fr

Mimoune MALKI
LabRi-SBA laboratory ,
Ecole supérieure en Informatique,
Sidi Bel-abbes, Algeria
m.malki@esi-sba.dz

***Abstract*—Industrial Cyber-Physical Systems (ICPS) have gained considerable interest in the last decade from both industry and academia. Such systems have proven particularly complex and provide considerable challenges to master their design and ensure their functionalities. In this paper, we intend to tackle some of these challenges related to the performance and decision supports of ICPS by proposing a federated learning based framework, called FedGA-ICPS. First, we initiate a ICPS modeling formalism to specify such systems structure and behaviors. Then, based on the ICPS generated data from the *industrial* sensors, FedGA-ICPS analyzes their performance by proposing locally embedded learning models. Then, federated learning is powered by genetic algorithm to accelerate and improve the aggregation. Finally, transfer learning is applied to broadcast the performed parameters of the leaning models over different constrained entities. FedGA-ICPS has been applied on MNIST and showed prominent results.**

***Index Terms*—Cyber-Physical Systems, Performance, Resilience, Federated Learning, Aggregation.**

I. Introduction

Over the past few years, IoT has become one of the most important technologies of the 21st century. There are multiple domains like the Internet of Industrial Things (IoIT) and the Internet of Medical Things (IoMT), which refer to the use of smart sensors, actuators, fast communication protocols, and efficient cybersecurity mechanisms to improve industrial and medical processes and applications. In large networks, smart devices generate large amounts of data, and thus IoT frameworks require intelligent, and robust techniques for big data analysis. Precisely, deep learning (DL) techniques have produced promising results in networks application due to their intelligent learning and processing capabilities. The traditional way to apply AI in CPS was centralized, such that there is a central server that trains the model using all the data from the end devices connected to it. However, transmitting such huge amount of data from these end-devices to the cloud lead to bandwidth congestion, data-processing delays, and potential leakage of privacy. Recently, this way of learning has been changed to the distributed learning or the federated leaning that groups together a set of clients (end devices) which have their own protected and non-shareable data. They collaborate with each other in order to have the same level of learning by sharing only the learning parameters.

Federated learning (FL) is a decentralized and a collaborative machine learning approach, introduced by Google in 2016. FL process has a purpose of generating a perfect model for a particular application. A typical workflow of its Life cycle of is to: choose the appropriate machine learning model to use select devices candidates to participate in the learning process where each device initializes its embedded model parameters and trains the model until its convergence. After a local training, all the connected devices upload their model parameters to the central server with a secure communication that we will discuss it later. Then, when the aggregator (central server for example) receives all local models, it will aggregate the parameters to update the new, global and optimal model. Again, the server re-shares the global model parameters and the devices update theirs parameters with the news. This process is repeated until the whole training process converges.

In the literature, McMahan et al. [1] proposed the FedAVG , federated average, where the weights of the different local models are averaged by the server to provide new weights and, thus, a new aggregated model. Also, FedPer proposed by Arivazhagan et al. [2] to the FedAVG in the way it computes new weights in the aggregated models [3]. However, the clients communicate the neural model's base layers to the server instead of the totality of the model and retain the other layers. The underlying idea is that the base layers deal with representation learning and can be advantageously shared by clients through aggregation. The upper layers are more concerned with decision-making, which is more specific to each client[3]. FedPer can be seen as an adaption of the transfer learning methodology into a federated learning scheme.The main goal of this work is to overcome the limitations of FedAVG and FedPer by introducing genetic algorithm to enhance the aggregation process of FL and deploy it for ICPS. The main contributions of this paper are as follows.

- Surveying the main contributions related to the application of FL in **CPS**.
- Developing a framework to enhance the performance analysis and the decision support to **ICPS**.
- Modeling formally **ICPS** components and their compositions.
- Proposing a federated solution to enhance the collaborative learning phases between **ICPS** components through genetic algorithms.
- Comparing **FedGA-ICPS** within the existing solutions and validates it on benchmarks.

II. Related Work

This section reviews and discusses approaches that deal with performance analysis and decision supports for **ICPS**.

Połap et al. [4] proposed an agent-based system to analyze and secure medical data that are collected from IoMT. Data are stored in a decentralized architecture where some are encrypted in a blockchain. The solution implements three agents: learning, indirect , and data management (DM). Learning agent (LA) launch six threads that train separately their proper CNN models for a specific part of the database. Indirect Agent (IA) classifies inputs coming from DM agent and communicates with the LA when the classification result is too low or uncertain to retrain the model. If the trained data are not enough, IA requests the data management agent (DMA) for completing the dataset with more entries. Unfortunately, this solution is not real-time since DMA takes time to get information from patients and doctors. Also, despite the use of FL methods, selecting the best classification method for this kind of noisy and heterogeneous data is challenging.

Tian et al. [5] proposed an asynchronous FL-based anomaly detection approach defined as a Delay Compensated Adam for resources constrained IoT devices using deep learning techniques. Their approach is established in three steps. First, pre-initialization of the parameters such that a random number of clients are selected for them to send a small part of their data to the server to train the global initial model. Second, they deployed an asynchronous model training. Although this approach is dedicated to anomaly detection, they did not study the reliability of the participating nodes. As the three-task server has run then it faces the challenges of bandwidth load and if it will crash then the system shuts down. Chen et al. [6] proposed FedHealth framework that applies federated transfer learning approach on cloud computing architecture for wearable healthcare. Their framework deals with the challenges of data isolation and model personalising in FL. Initially, the cloud server constructs the global model using public datasets, distributes it to the clients via homomorphical encryption, then the clients retrain local models and upload them to the cloud server. Repetitively, the aggregator updates the global model using FedAVG algorithm, distributes it to the clients and perform transfer learning.

Hao et al. [7] proposed Privacy-enhanced FL (PEFL) scheme for Industrial Artificial Intelligence to increase the security of model gradients shared between the server and clients and ensure a good accuracy for local models. Their system architecture consists of (1) Key generation center (KGC) to distribute private keys to each participant, (2) CS is the aggregator based on Cloud and (3) participants. First, each participant learns local model, calculates local gradients and perturbs these by adding local noises using Differential Privacy (DP) technique. The perturbed gradients are encrypted into the BGV ciphertext using Homormorphic encryption and the ciphertext BGV encrption is embedded into augmented learning. Then, the CS performs the reverse process of encryption by executing a set of decryption steps for aggregation.The results show that when only a small percentage of participants are affected by the adversary, accuracy decreases little. Zhou et al. [8] proposed a privacy preserving federated learning scheme in fog computing. Acting as a participant, each fog node is enabled to collect IoT device data and complete the learning task. Such design effectively improves the low training efficiency and model accuracy caused by the uneven distribution of data and the large gap of computing power. They enabled IoT device data to satisfy differential privacy to resist data attacks and leverage the combination of blinding and Paillier homomorphic encryption against model attacks, which realize the security aggregation of model parameters.

Saha et al. [9] proposed a FogFL framework to implement FL approach on Fog Computer layer in order to reduce communication latency and energy consumption of resource-constrained edge devices and increase system reliability using heuristic approach for selecting an optimal fog node as a global aggregator at each FL iteration. Their system is designed as a set of edge devices clusters, each cluster is related to a closer fog node. After the local aggregation, each fog node send workload and communication latency parameters to the cloud server. Unfortunately, all nodes (cloud/ fog) are centralized that could affect the FL approach. Wang et al. [10] proposed an environment monitoring of air quality framework based on fog computing that uses FL approach with multi-source heterogeneous collected data. Their architecture is composed of IoTs, Edge Nodes and Fog Gateway Device (FGD) that represent the Local Multi-source heterogeneous data Fusion System (LMFS) and Cloud Center that represents the Centralized Homologous data Training System (CHTS). For LMFS, they deployed five sub-classifiers on the fog node, each of them has one or more heterogeneous data datasets that has pre-processing first and a shared task, and they extracted the numerical features from all sub-classifiers on edge node to get the local assessment result.

Yao and Ansari [11] proposed a FL enhancement approach based on Fog Computing to accelerate FL time and minimize the energy consumption by controlling the CPU frequency and wireless transmission power (WTP) of all IoT devices. FL time consists of the computation time for local model training and the wireless transmission time for uploading local model updates to the fog node that should satisfy the QoS requirement (it has not to be more than the maximum FL time). It is the same of FL energy consumption. To calculate the optimized WTP and CPU frequency values of an IoT device, they implemented an alternative direction algorithm (ALTD) within each local iteration in all the IoT devices. Qu et al. [12] proposed A Framework for Cognitive Computing in Industry 4.0 Networks by developing a decentralized paradigm based on Blockchain-enabled FL to improve the performances of Industry 4.0 manufacturing by securing the data, performing efficient processing, providing incentive mechanisms to contribute to the learning, and avoiding poisoning attacks. Each device in the network has its private data that is stored locally and used to train models by FL, this mechanism ensures the security of the data and efficient processing by sharing models rather than the raw data. Instead of deploying a central server that aggregates models shared by end-devices, they proposed a blockchain architecture with public ledgers to fully decentralize FL with the proof-of-work (PoW) consensus

algorithm by replacing it with a selected temporary aggregator in each round, so the parameters of the updated models of end-devices are sent to the cluster of miners that are end-devices.

Unfortunately, the discussed solutions do not consider data and models heterogeneity as well as the models convergence is not satisfactory of **ICPS**. However, with respect to the surveyed approaches, the proposed solution initiates a step to improve the aggregation performance in federated learning for **ICPS**. At this level, we focus more on the precision of the aggregation and the application of FL on **ICPS** by developing a complete federated learning framework, called **FedGA-ICPS**.

III. FEDGA-ICPS FRAMEWORK

As shown in Figure 1, **FedGA-ICPS** develops five stages: CPS (red rectangle), Learning (blue rectangle), Eletion (green rectangle), Aggregation (yellow rectangle), and Broadcasting (violet rectangle). In the following, we explain each part and steps of **FedGA-ICPS**.

(1) Initially, **FedGA-ICPS** develops and implements a **CPS** as a composition of entities and components of different forms and nature. Each entity has its proper structure and behavior. The entities can communicate and interleave in different environment.

(2) Then, through simulation and run-time execution , **FedGA-ICPS** collects, formats, cleans and normalizes streaming data " generated and communicated between different **CPS** components. " will be used for the learning step that relies on a convolution neural network that is assigned to a given device.

(3) Consequently, **FedGA-ICPS** proposes a set of component to elect the best candidate to federate the learning between the embedded CNN. The election takes into consideration different parameters, like: processing and memory capacities, latency, availability, security, etc.

(4) After a local convergence learning, **FedGA-ICPS** perform the aggregation through genetic algorithms. The latter takes into consider: the weights of the local models. Then, produced in the elected component the optimal weight vector for broadcasting.

(5) Finally, **FedGA-ICPS** broadcast to the different clients, edges, and components the resulting optimal weights through transfer parameter learning.

The following section details each step of **FedGA-ICPS**.

IV. INDUSTRIAL CPS

We consider a system $\mathcal{S}$ as a composition of a set of entities $\mathcal{E}$ that interact and interleave through a network of physical and logical channels ($\mathcal{N}$) to accomplish a given task ($\mathcal{T}$). A system $\mathcal{S}$ is a tuple $\langle \mathcal{E}, \mathcal{N}, \mathcal{T} \rangle$.

A. Modeling the entities

An entity $\varepsilon \in \mathcal{E}$ can be an IIoT, an edge node, a fog node or a cloud server that are enabled to execute specific actions, or collaborate with other entities to form a system executing a global task. To evaluate the guard related to an action, the entity ε run its associated machine learning $_{\varepsilon}$ that evaluates the variable of the specified guard. To enhance the decisions of an entity ε, **FedGA-ICPS** develops different learning mechanisms each period of time. However, ε is the main entities describing **ICPS**, and it is defined by the tuple $\langle id, attr, Actuator, \mathcal{V} \rangle, \Sigma, Beh$, where:

- id is a finite set of tags,
- $attr : id \rightarrow 2^T$ returns the attributes of an entity,
- $Actuator$ specifies the status of an entity by evaluating its attributes,
- $\mathcal{V}$ is a set of local variables that can be used as parameter in the entity, such as a sensing value, a dataset or a model.
- $\Sigma = \{\texttt{Send}, \texttt{Receive}, \texttt{Update}, \texttt{Predict}, \texttt{Train}, \texttt{Aggregate}\}$ is a finite set of atomic actions that depend on the type of entity ε_i and executed by the latter.
- $Beh\colon id \times \Sigma \rightarrow \mathcal{L}$ returns the expression written in the language $\mathcal{L}$ that describes the behavior of an entity. The syntax of $\mathcal{L}$ is given by: $B ::= \alpha \mid B \cdot B \mid B +_g B$, where $\alpha \in \Sigma$, "$\cdot$" composes sequentially the actions and $+_g$ is a guarded choice decision that depends on the evaluation of the guard, a propositional formula, g, by the functionality $\texttt{Predict}$. When $g \triangleq \top$ the guarded decision become a non-deterministic choice.

B. Modeling the Network

The network $\mathcal{N}$ defines how the entities are connected and communicate. An entity ε_i can be connected to another one through a physical or logical channel for communication or to a subsystem. In $\mathcal{N}$.

We define a network $\mathcal{N}$ as a graph where vertices are the entities and the edges are the way that they interact and connected $\mathcal{N} = \langle \mathcal{E}, \texttt{Chan}, \texttt{Prot}, \texttt{Rel} \rangle$, where:

- $\texttt{Chan}$ is a finite set of channels,
- $\texttt{Prot}$ is a finite set of protocols where $\epsilon_{\texttt{Prot}}$ is the empty protocol.
- $\texttt{Rel} : \mathcal{E} \times \mathcal{E} \rightarrow \texttt{Chan} \times \texttt{Prot}$ relies two entities with a channel and a protocol. When $\epsilon_{\texttt{Prot}}$ is assigned, it means both nodes are physically connected.

Fig. 2 shows a physical relation between ε_i and ε', and a logical relation through the protocol $Prot$ between the entities ε_i and ε.

C. Modeling the Tasks

The task $\mathcal{T}$ is the main goal of the system. It describes the sequence of actions that should be realized by each entity. We define a task by a tree where the root represents the main goal of the system $\mathcal{S}$, the children are sub-goals of the entities, and leafs are the final product for each entity. The task $\mathcal{T}$ is the tuple $\langle \texttt{Goals}, \preceq \rangle$, where:

- $\texttt{Goals}$ is a finite set of goals where $\texttt{G} \in \texttt{Goals}$ is the root (the main goal),
- $(\texttt{Goals}, \preceq)$ is a preorder relation on $\texttt{Goals}$.

V. LEARNING

In the learning step, **FedGA-ICPS** relies on CNN, convolutional neural network, which is a class of deep neural network [13]. A given CNN is learned in a given machine locally by considering only local data. As depicted in Fig. 3, the architecture of CNN consists of three layers: (1) convolution, (2) pooling, and (3) fully connected. The convolution operation

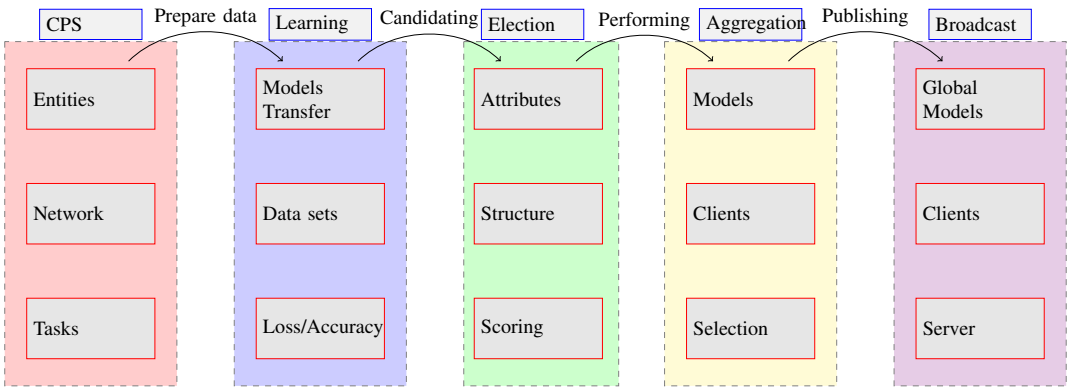

Fig. 1. The Interoperability and Integrity Validation and Evaluation.

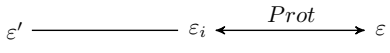

Fig. 2. A Network of logical and physical relation between the entities.

using multiple filters is able to extract features (feature map) from the data set, through which their corresponding spatial information can be preserved. Pooling is to reduce the dimensionality of feature maps from the convolution operation. Max pooling and average pooling are the most common pooling operations used in the CNN [14]. The Fully-Connected Layer is responsible for categorization using the features extracted by the preceding layers. While convolutional and pooling layers often utilize ReLu functions to classify inputs, FC layers typically use a softmax activation function to provide a probability from 0 to 1.

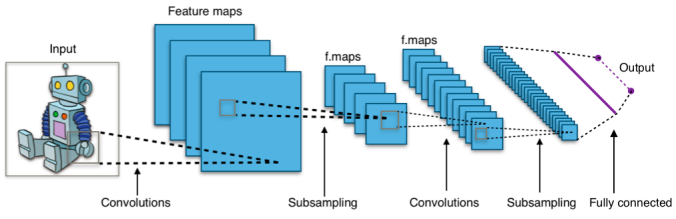

Fig. 3. CNNs Architecture.

Zhang and Sabuncu[15] declared that the most common way to train CNNs for classification problems is to use stochastic gradient descent coupled with the Cross Entropy (CE) loss.

VI. Election

To elect the appropriate candidate for a federated learning, **FedGA-ICPS** looks for the most powerful component. As default, we consider the computing cloud server as the aggregator, then, depends on the free memory of a component and its processing capacity, the cloud server can elect it as a secondary aggregator. **FedGA-ICPS** reorders all S components as a list of aggregators with priorities.

VII. Aggregation

Compared to FedAVG and FedPer, FedGA communicates the elected aggregator only the base layer. Then, the latter calculates the new weigths calls the genetic algorithm (line 9) that runs as follows.

(1) Define an adequate *chromosome* which the weight vector.
(2) Select a large set of chromosomes, *population* takes into account all weight vectors collected from the different components.
(3) Apply the reproduction operators (*selection, crossover, mutation*). *selection* is applied on on vectors with high ranking (fitness evaluation). In our case, the fitness is the loss function. The *crossover* operation is based on the single point paradigm. It means that a vector is divided into two parts to be exchanged with another vector to form a new population. Finally, the *mutation* operation collects randomly only 10% of weights to reproduce new vectors.
(4) **FedGA-ICPS** repeats this process until the accuracy is more than 99%.

In FedGA, a weight vector is used to represent the system throughout the genes.

VIII. Broadcasting

After the aggregation phase where a new model is obtained with a high accuracy and low errors in test and training, **FedGA-ICPS** communicates this model to participated components in the learning phase. Again, to exploit better the resulting model, we update the local models using transfer learning. Unlike the traditional machine learning paradigm where the learning process happens in isolation, without considering knowledge from any other domain, transfer learning uses knowledge from other existing domains (source) during the learning process for a new domain (target). To deal with transfer learning concepts [16], we define a domain $\mathcal{D}$ contains two elements, feature space, χ, and marginal probability,$P(X)$, where $X = \{x_1, x_2, ..., x_n\} \in \chi$ is a sample data point. Thus, they represent the domain mathematically as $D = \{\chi, P(X)\}$.
A task $\mathcal{T}$ can be defined as a two-element tuple of the label space, $\mathcal{Y}$, and objective function $f : \chi \rightarrow \mathcal{Y}$. The function f is used to predict the corresponding label $f(x)$ of a new instance x. This task, denoted by $\mathcal{T} = \{\mathcal{Y}, f(x)\}$, is learned from the training data consisting of pairs $\{x_i, y_i\}$, where

Algorithm 1 Federated Genetic Algorithm (FedGA), $\mathcal{C}$ is the fraction of clients selected to participate in each communication round. The **K** clients are indexed by **k**; **B** is the local mini-batch size, $\mathcal{P}_k$is the dataset available to client **k**, $\mathcal{G}$ is the dataset used for the test which is available on the server, $W_{\mathbf{B}}$ is the vector of base layers, where $W_{\mathbf{P}}$ is the vector of personalized layers, **E** is the number of local epochs, and η is the learning rate.

Procedure FedGA ▷ run on the server
Initialize $W_{\mathbf{B}}^{0}$
for `each round` $\mathbf{t} = 1, 2, 3, \ldots$ **do**
  $m \leftarrow max(C.K, 1)$
  $S_t \leftarrow$ `(random set of` **m** `clients)`
  for `each client` **k** $\in S_t$ **do**
    $W_{\mathbf{B,k}}^{t+1} \leftarrow ClientUpdate(\mathrm{k}, \mathrm{W}_{\mathbf{B,k}}^{t})$ ▷ In Parallel
  end for
  $W_{\mathbf{B}}^{t+1} = GA(\mathcal{G}, W_{\mathbf{B}}^{t})$ ▷ Only base layers are aggregated
end for
End procedure FedGA
Procedure ClientUpdate(k, w_B^t) ▷ run on client **k**
$\beta \leftarrow (Split\mathrm{P}_k$into mini-batches of size B)
for each local epoch **i** from 1 to **E** **do**
  for batch **b** $\in \beta$ **do**
    $w_{\mathbf{B}}, w_{\mathbf{P}} \leftarrow w - \eta \Delta \mathcal{L}(w_{\mathbf{B}}, w_{\mathbf{P}}, b)$ ▷ base layers are updated and trained and personnalized layers are trained
  end for
end for
`return` **t** `to the Server`
End procedure ClientUpdate$(k, w_{\mathbf{B}})$

$x_i \in X$ and $y_i \in \mathcal{Y}$. It can also be denoted as $P(Y|X)$ from a probabilistic view point. Given a source domain $\mathcal{D}_S$ a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$ [17]. The Figure 4 demonstrates the use of Transfer Learning in our Framework, with only the convolutional layers being transferred.

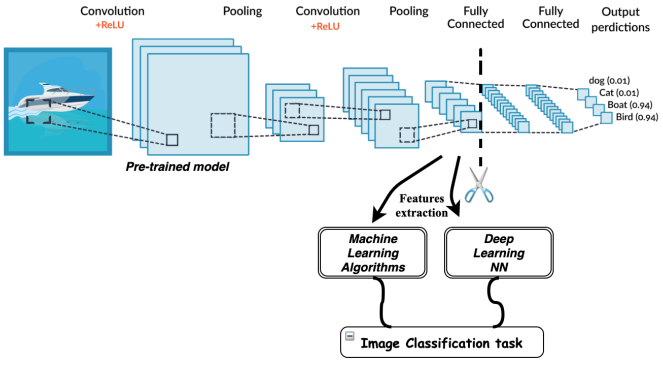

Fig. 4. Transfer Learning Architecture.

IX. Experimental Results

We evaluated the performance of our **FedGA-ICPS** framework using the MNIST dataset [18]. We divide it into heterogeneous subsets. Each client has a different part to other clients with an unequal size (non-iid data). We use a window-frame size of 1000 samples. Our experiments were done using CNN to compare the federated learning combined with genetic algorithm results against FedAVG and FedPer. Our CNN model has two convolutional layers followed by a max-pooling layer where the outputs are fed to two fully-connected layers. The models are trained using a mini-batch SGD of size 1000, ReLU activation function and to counter over-fitting, a dropout is used. The models were developed using Pytorch for our implementations. Using FedAVG as an aggregation algorithm, the results presented in Figure 5 show that is a perturbation of the accuracy for each customer due to data heterogeneity and dataset size variance.

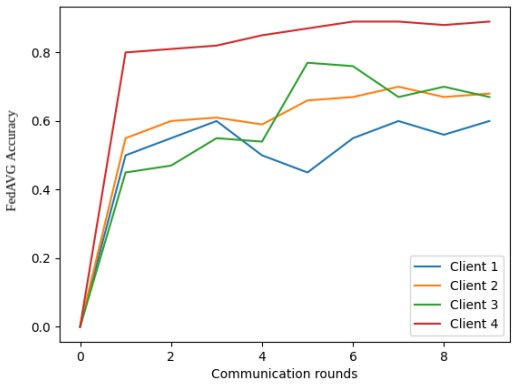

Fig. 5. FedAVG Aggregation Algorithm.

Using FedPer, the model is split in base and personalized layers. Personalized layers are not communicated to the server, only the base layers are aggregated by the federated server, using transfer learning. For the two-layered CNN used in this study, the last dense layers are the personalized layer. This means that it is not communicated to the server, only the lower layers are trained using the FL approach. The results show that is a perturbation of the accuracy for each customer due to data heterogeneity and dataset size variance. The results depicted in Figure 6 present that there is an improvement in accuracy compared to FedAVG. Finally, using **FedGA-ICPS**, we obtain a rapid convergence for all clients with an average accuracy greater than 98% as shown in Figure 7.

X. Conclusion

We have given a first step toward a comprehensive methodology for enhancing performance analysis in order to create a more robust **ICPS** in this study. The developed **FedGA-ICPS** framework describes a system as a collection of things, each of which has its own structure and behavior for carrying out a certain purpose. To expedite the analysis and learning processes, **FedGA-ICPS** intends to use federated learning and genetic algorithms to assist limited devices in locally embedding their decision models. We demonstrated the efficacy of

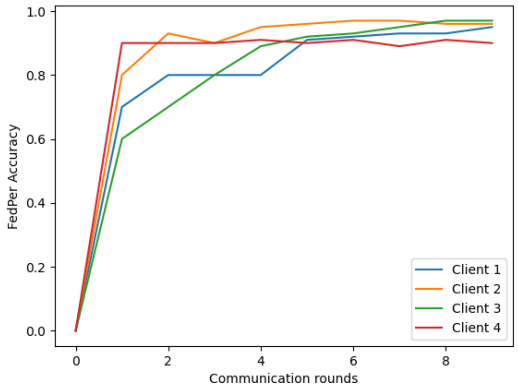

Fig. 6. FedPer Aggregation Algorithm.

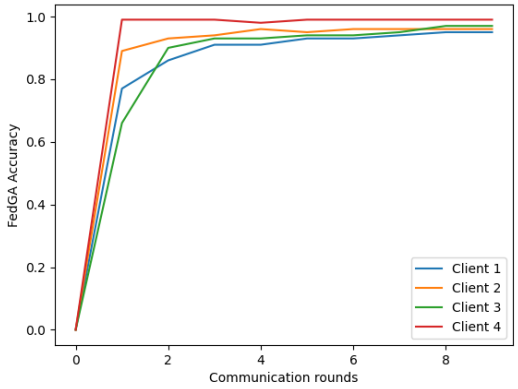

Fig. 7. FedGA Aggregation Algorithm.

the suggested framework using a renowned benchmark. We intend to expand the framework's capabilities in the future by (1) incorporating additional machine learning techniques and automatically selecting the best agent mostly through reinforcement learning, (2) decentralizing the system through a blockchain architecture, and (3) evaluating the framework on more complex use cases and benchmarks.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[3] S. Ek, F. Portet, P. Lalanda, and G. Vega, "A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison," in *19th IEEE International Conference on Pervasive Computing and Communications PerCom 2021*, 2021.

[4] D. Połap, G. Srivastava, and K. Yu, "Agent architecture of an intelligent medical system based on federated learning and blockchain technology," *Journal of Information Security and Applications*, vol. 58, p. 102748, 2021.

[5] P. Tian, Z. Chen, W. Yu, and W. Liao, "Towards asynchronous federated learning based threat detection: A dc-adam approach," *Computers & Security*, vol. 108, p. 102344, 2021.

[6] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.

[7] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2019.

[8] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10 782–10 793, 2020.

[9] R. Saha, S. Misra, and P. K. Deb, "Fogfl: Fog-assisted federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8456–8463, 2020.

[10] W. Wang, C. Feng, B. Zhang, and H. Gao, "Environmental monitoring based on fog computing paradigm and internet of things," *IEEE Access*, vol. 7, pp. 127 154–127 165, 2019.

[11] J. Yao and N. Ansari, "Enhancing federated learning in fog-aided iot by cpu frequency and wireless power control," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3438–3445, 2020.

[12] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Trans. on Ind. Informatics*, vol. 17, no. 4, pp. 2964–2973, 2020.

[13] S.-C. Huang and T.-H. Le, "Chapter 4 - multi-category classification problem," in *Principles and Labs for Deep Learning*, S.-C. Huang and T.-H. Le, Eds. Academic Press, 2021, pp. 81–116.

[14] W. Zhu, Y. Ma, Y. Zhou, M. Benton, and J. Romagnoli, "Deep learning based soft sensor and its application on a pyrolysis reactor for compositions predictions of gas phase components," in *13th International Symposium on Process Systems Engineering (PSE 2018)*. Elsevier, 2018, vol. 44, pp. 2245–2250.

[15] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[16] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[17] Y.-P. Lin and T.-P. Jung, "Improving eeg-based emotion classification using conditional transfer learning," *Frontiers in human neuroscience*, vol. 11, p. 334, 2017.

[18] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.