Aggregation using Genetic Algorithms for Federated Learning in Industrial Cyber-Physical Systems

Souhila Badra Guendouzi, Samir Ouchani, Mimoun Malki

Aggregation using Genetic Algorithms for Federated Learning in 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

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

***Abstract*—Industry and academics are interested in industrial cyber-physical systems (`ICPS`). Complexity makes it hard to grasp these systems design and functioning. By offering `FedGA-ICPS`, a federated learning framework based on genetic algorithms, we can solve `ICPS`ś performance and decision support. To simulate the structure and behavior of these systems, we use `ICPS`. `FedGA-ICPS` investigates the performance of the `ICPS` sensors by offering locally integrated learning models. Genetic algorithm then speeds up and improves federated learning aggregation. Transfer learning is used to disseminate model parameters across restricted entities. Fashion MNIST's initiative achieved notable outcomes.**

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

I. Introduction

IoT is one of the most crucial 21st-century technologies. Internet of Industrial Things (IoIT) and Internet of Medical Things (IoMT) utilize smart sensors, actuators, rapid communication protocols, and efficient cybersecurity procedures to enhance industrial and medical processes and applications. Clever devices create a lot of data in vast networks, hence IoT frameworks need smart, resilient big data analysis tools. Deep learning (DL) algorithms have shown promise in network applications owing to their intelligent learning and processing. Traditional AI in CPS was centralized, with a central server training a model using data from linked end devices. Transmitting so much data from end-devices to the cloud causes bandwidth bottlenecks, processing delays, and privacy leaks. This form of learning has recently been altered to distributed or federated learning, which groups clients (end devices) with their own secured and non-shareable data. They share learning parameters to achieve the same degree of learning.

Google announced decentralized, collaborative, and federated learning (FL) in 2016. FL creates a model for a certain application. Choose the proper machine learning model to utilize, select devices on which candidates will be able to learn, and train the model until convergence. The central server receives the model parameters from all linked devices after local training has been completed. The aggregator (central server) aggregates the parameters of all local models before it updates the global model when it receives all local models. The server re-shares global model parameters, and devices update theirs. This is performed until training converges.

In the literature, Brendan McMahan et al. [1] proposed the FedAVG algorithm, which combines the weights of several local models to produce a new weight and an aggregated model based on the average. Similarly to FedAVG, FedPer proposed by Arivazhagan et al. [2] uses an aggregated model to calculate the new weights used in the aggregation process [3]. The clients only send the neural model's base layers to the server, keeping the rest. Base layers deal with representation learning and may be aggregated by clients. Upper levels focus on client-specific decision-making [3]. This study aims to overcome FedAVG and FedPer's limitations by using a genetic algorithm to improve FL aggregation for `ICPS` . As shown in Figure 1, `FedGA-ICPS` develops five stages: ICPS (red rectangle), Learning (blue rectangle), Election (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 `ICPS` as a composition of entities and components of different forms and nature. Structure and behavior are unique to each entity. Communication and interleaving are possible between the entities in a variety of environments.
(2) Then, through simulation and run-time execution , `FedGA-ICPS` collects, formats, cleans and normalizes streaming data " generated and communicated between different `ICPS` 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` broadcasts to the different edges, and components the resulting optimal weights through transfer learning.

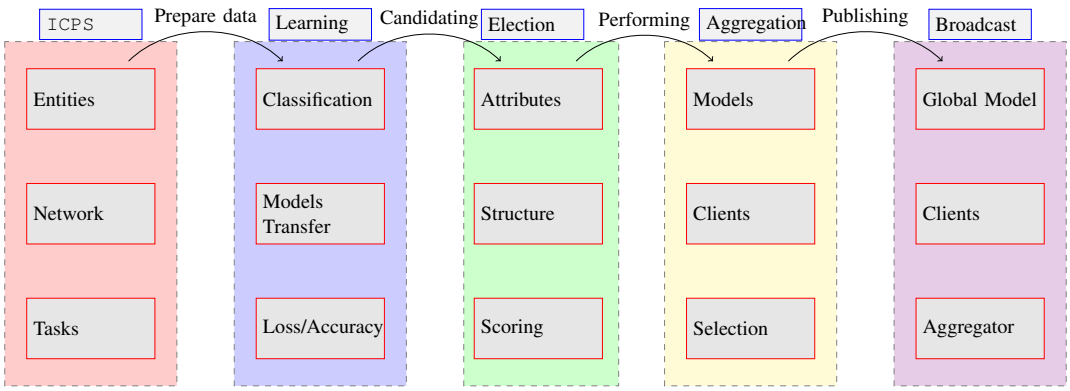

Fig. 1. Aggregation using Genetic Algorithms Approach for Federated Learning in Industrial Cyber-Physical Systems.

In a nutshell, we present our main contributions.

- Surveying the main contributions related to the application of FL on CPS.
- Develop a framework called FedGA-ICPS to enhance the performance to ICPS.
- Model formally ICPS components and their compositions.
- Propose a federated solution to enhance the collaborative learning phases between ICPS components through genetic algorithms.
- Compare FedGA-ICPS within the existing solutions and validates it on benchmarks.

The remainder of this paper is organized as follows. Section II surveys contributions related to federated learning and its applications. Section III develops our proposed framework, FedGA-ICPS that is validated in IV. Finally, Section V concludes the paper and give hints on our FedGA-ICPS related perspectives.

II. Litterature

The purpose of this section is to examine and discuss approaches that deal with performance analysis and decision support for ICPS.

Połap et al. [4] proposed an agent-based system to evaluate and secure IoMT medical data. Decentralized data are encrypted on a blockchain. Learning, indirect, and data management are actors in the solution (DM). Learning agent (LA) launches six threads to train CNN models for a specified database section. Indirect Agent (IA) classifies DM agent inputs and notifies LA when results are too low or unclear to retrain the model. If the training data are insufficient, IA asks additional entries from the DMA. DMA takes time to gather patient and doctor information, thus this solution isn't real-time. Despite FL techniques, choosing the appropriate classification method for noisy and diverse data is difficult.

Tian et al. [5] used methods from deep learning to come up with a proposal for an asynchronous FL-based anomaly detection method that they described as a Delay Compensated Adam for use with IoT devices that have limited resources. Their strategy is built up in stages, beginning with step one. First, there will be a pre-initialization of the parameters in such a way that an arbitrary number of clients will be chosen for the purpose of having them submit a tiny portion of their data to the server. As a result, the initial global model will be trained on the server. Second, they implemented an asynchronous training method for the model. They did not evaluate the dependability of the participating nodes, despite the fact that this method is specifically designed for anomaly identification. As soon as the three-task server has started running, it is confronted with the difficulties of a high bandwidth demand, and if it crashes, the whole system is terminated. As part of wearable healthcare, Chen et al. [6] proposed federated transfer learning to be integrated with cloud computing. Their framework handles FL data separation and model personalization. The cloud server builds a global model using public datasets, distributes it to clients through homomorphic encryption, and clients retrain local models and upload them. The aggregator refreshes the global model using fedAVG and does transfer learning..

Hao et al. [7] presented Privacy-enhanced FL (PEFL) for Industrial AI to promote model gradient security and local model accuracy. (1) Key generation center (KGC) distributes private keys to each participant, (2) CS is a Cloud-based aggregator, and (3) participants. Each participant learns a local model, produces local gradients, and perturbs them using Differential Privacy (DP). The perturbed gradients are encrypted into BGV ciphertext using Homormorphic encryption. The CS reverses encryption by decrypting for aggregation. When a small fraction of participants are impacted, accuracy is barely affected. Zhou et al. [8] proposed a fog computing federated learning approach with privacy protection as a means of protecting privacy. In order to complete the learning process, each fog node collects and analyzes data from IoT devices. Due to unequal data distribution and computational resource gaps, this strategy increases the efficiency of training and improves model accuracy due to the unequal distribution of data. Using blinding and Paillier homomorphic encryption to protect the parameters of a model and deploying differential privacy to resist data attacks on IoT devices, they are able to resist data attacks on IoT devices.

Wang et al. [9] suggested the use of fog computing to monitor air quality using a multisource heterogeneous dataset that uses a FL approach with a multi-source approach using multi-source heterogeneous data collection. IoTs, Edge Nodes,

and Fog Gateway Devices (FGD) make up the architecture of the Local Multi-source Heterogeneous Data Fusion System (LMFS) and the Centralized Homogeneous Data Training System (CHTS), which represents the centralized homogeneous training system of IoTs, Edge Nodes, and Fog Gateway Devices. LMFS was deployed with five sub-classifiers on the fog node, each of them has one or more heterogeneous datasets that needs pre-processing first, followed by a shared task, and the numerical features are extracted from all five layers on the edge node to get the local assessment results. Yao and Ansari [10] used fog computing to accelerate FL time and minimize the power consumption of its IoT devices, he proposed a new FL enhancement method that uses CPU frequency control and wireless transmission power (WTP) control to simplify FL and accelerate FL time and minimize energy consumption. There is a requirement for the FL time to be less than the maximum FL time, since the FL time includes the computation time for the local model training and the wireless transmission time for uploading the local model updates to the fog node that should satisfy the QoS requirement. It is the same with regard to the energy consumption of FLs. In all IoT devices that have been tested, they have implemented an alternative direction algorithm (ALTD) within each local iteration in order to calculate the optimal WTP and CPU frequency values.

Regarding survey approaches, the proposed solution improves federated learning aggregation for ICPS . At this stage, we concentrate on the accuracy of FL aggregation and application on ICPS by developing FedGA-ICPS a federated learning framework.

III. FedGA-ICPS Framework

This section follows FedGA-ICPS steps depicted in Fig. 1 by firstly presenting ICPS formalism. has been applied on Fashion MNIST.

A. Industrial CPS

In our approach, a system $\mathcal{S}$ is composed from a set of nodes (atomic entities) $\mathcal{E}$ that communicate and interact over a network composed from physical and logical channels ($\mathcal{N}$) to ensure that a given task is completed ($\mathcal{T}$) in a specific context $\mathcal{CT}$. We formulate a given system $\mathcal{S}$ by the tuple $\langle \mathcal{E}, \mathcal{N}, \mathcal{T}, \mathcal{CT} \rangle$ detailed as follows.

1) Entities formalization: A given 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 organize a global task-execution system in collaboration with other entities. To evaluate the guard related to an action, the entity ε run its associated machine learning $\texttt{ml}_\varepsilon$ that evaluates the variables of the specified guard. To enhance the decision-making of an entity ε, FedGA-ICPS develops techniques to help entities to update periodically their associated mls. However, the atomic entity ε that describes ICPSis stipulated as $\langle id, attr, Actuator, \mathcal{M}, \Sigma, Beh \rangle$, where:

- id is a closest set of identifiers,
- $attr : id \rightarrow 2^T$ is a function associating the possible attributes to a given identified entity,
- $Actuator$ evaluates the attributes of an entity to set its status,
- $\mathcal{M}$ is a function that associate to an entity ε its ml.
- $\Sigma = \{\texttt{Sending}, \texttt{Receiving}, \texttt{Updating}, \texttt{Predicting}, \texttt{Training}, \texttt{Aggregating}\}$ is the finite set of atomic actions that depend on the type of entity ε_i and executed by the latter.
- $Beh\colon id \times \Sigma \rightarrow \mathcal{L}$ sets the behavior of an entity ε_i expressed in the language $\mathcal{L}$. The BNF of $\mathcal{L}$ is described by: $B ::= B +_g B \mid \alpha \mid B \cdot B$, where $\alpha \in \Sigma$, " $\cdot$ " is sequential composition operator, and $+_g$ is a deterministic choice operator relying on the guard evaluation. The atomic propositions g is evaluated by Predicting. The case when no propositions assigned to g, we consider $g \triangleq \top$. Thus, the deterministic decision become a non-deterministic.

2) Networking Composition: By using this network $\mathcal{N}$, the entities can communicate and establish a connection between each other in a safe and secure environment. In order to communicate with another entity ε_i or to a subsystem, an entity ε_i must be connected to it through either a physical or logical channel. Networks are defined as graphs with vertices representing entities and edges representing the way in which they interact and are connected. A network $\mathcal{N}$ is defined as the combination of entities, channels, protocols, and relations. It is represented by the tuple $\langle \mathcal{E}, \texttt{Chan}, \texttt{Prot}, \texttt{Rel} \rangle$, where:

- Chan is a finite set of secure and accessible channels that can be accessed,
- Prot is a range of protocols compatible and classified for nodes and channel where $\epsilon_{\texttt{Prot}}$ denotes the empty protocol.
- $\texttt{Rel} : \mathcal{E} \times \mathcal{E} \rightarrow \texttt{Chan} \times \texttt{Prot}$ links two entities through the use of a channel and a protocol. The assigned value of $\epsilon_{protocols}$ indicates that the two nodes have a physical connection between them.

3) Tasks Modeling: In order to accomplish the main function of the system, the task $\mathcal{T}$ is the primary objective. As a result, each entity is expected to realize a specific sequence of actions to accomplish its goals.Our definition of a task is based on a tree, if the root of the tree represents the main goal of the system $\mathcal{S}$, the children represent the sub-goals of each entity, and the leaves represent the final result of each entity. The task $\mathcal{T}$ is represented by the tuple $\langle \mathcal{G}, \preceq \rangle$, where:

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

4) Context: It can be seen as a container of entities that can change dynamically, by integrating or excluding entities, changing protocols, and updating tasks, but they should follow certain rules and policies $\mathcal{PL}$. A context $\mathcal{CT}$ is the tuple $\langle \mathcal{E}' \subseteq \mathcal{E}, \mathcal{T}' \subseteq \mathcal{T}, \mathcal{PL} \rangle$. In FedGA-ICPS, a policy is expressed as a temporal logic formula.

B. Learning

The creation of a local model for each edge item is the goal of this step of the process. It is made up of two separate processes working in tandem. Specifically, the following is what should be done:

1) Classification: CNN, short for convolutional neural network, is a kind of deep neural network [11] that is used in the learning phase of FedGA-ICPS. This stage depends on CNN. A specific CNN is machine-learned in that machine by evaluating just the data that pertains to that CNN. The architecture of a CNN is represented in Figure 2, which shows that the architecture is composed from the convolution layer, the pooling layer,as well as the fully-connected layer. The dataset is capable of having its spatial information preserved when the convolution process with many filters is applied to it. This results in the extraction of features (feature map). Convolution is a method for creating feature maps, while pooling technique is to reduce the size of the resulting feature maps. Often, Two types of pooling are widely used in CNN [12]: *maximum* and *averaging*. The fully-connected layer is in charge of classifying the data into several categories according to the attributes that were gathered by the layers that came before it. In the convolutional and pooling layers of a neural network, ReLu functions are often employed to classify the inputs. On the other hand, in the FC layers, a softmax activation function is typically utilized to give a probability ranging from 0 to 1. Because it is reliant on the structure of its own dataset, each edge client in our system has its own CNN model. This model does not have to be similar to other local models in terms of the number of filters, layer design, function activation, and so on.

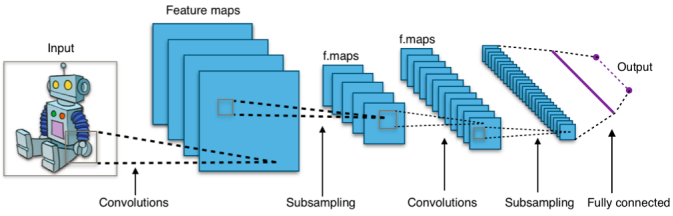

Fig. 2. CNNs Architecture.

At training stage, Zhang and Sabuncu[13] mentioned that integrating stochastic gradient descent with the Cross Entropy was the most popular method used to train CNNs for classification issues (CE).

2) Models Transfer: Transfer learning, also known as TL, is the practice of reusing a model that has already been trained on another learning task that is related to the first. A deep neural network requires a great deal of data and resources along with a lot of computing power to make good decisions, and hence transfer learning is recognized as a solution that can help to overcome these challenges by using data and resources in order to accelerate the learning process. Because the majority of practical applications do not need millions of labeled samples to train intricate deep learning models, this is especially useful in industrial settings. In fact, this is particularly effective in industrial contexts. The mechanism of transfer learning may be used in two different ways: (1) This procedure involves taking a model that has already been trained on a source dataset and training it again on a target dataset to improve its performance. This process is called fine-tuning pre-trained models. Then, (2) Performing fine-tuning on the feature extractors, which involves freezing the feature extractor layers and just retraining the classification layers. In the course of our research, we concentrate on the process of fine-tuning the feature extractors presented in Figure 3.

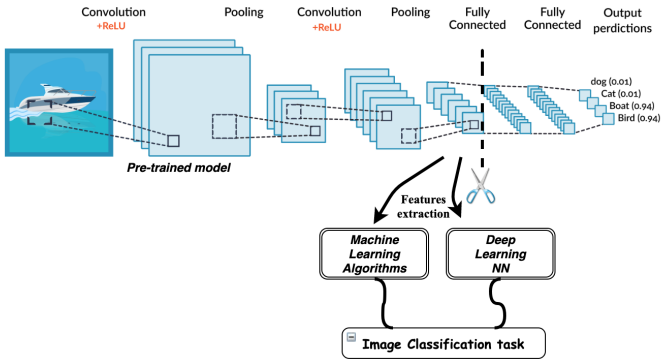

Fig. 3. Transfer Learning Architecture.

When a new edge node is added to the industrial network, the cloud server searches for pre-trained models that are comparable to those already deployed on other nodes of the same kind. These models must have been trained using a large dataset and must have reached a high level of accuracy. As a consequence of this, only the base layers of the network need to be retrained as opposed to the whole network, since the feature extractor layers are the only ones that are sent to the target node.

C. Election

For FL, FedGA-ICPS seeks for the most powerful Fog or Edge node. As default, we consider the cloud server as the aggregator. Depending on a component's free memory and processing capabilities, the cloud server may elect it as a secondary aggregator. FedGA-ICPS reorders all system components into a prioritized list.

We have formulated the election process by an election tree $\mathcal{ET}$, which is characterized by the tuple $\langle \mathcal{E}, \mathcal{C}, \mathcal{P} \rangle$, where:

- $\mathcal{E}$ are the entities that are detailed in Section III-A1.
- $\mathcal{C}$ returns a value that takes into consideration an entity attributes as its actual available capacity memory, processing capacity, etc.
- $\mathcal{P}$ assigns a priority value to a given entity ε to resolve the non-determenistic problem when entities capacities are equals.

FedGA-ICPS considers to measure periodically the capacity of each node that can change the election order. As example, in Fig 4, the cloud server ε_1, which is the most powerful entity at the moment $t = 0$, is represented by the root of the tree $tree$. Children are the least efficient entities, whereas leaves, which are often IoT devices, are the weakest. Each entity's priority $\mathcal{P}_i$ functions are represented by the arcs.

D. Aggregation

Unlike FedAVG and FedPer, FedGA simply uploads encrypted base layer (classification layer) weights to the designated aggregator. The latter creates the new weights by executing the genetic algorithm (line 7) and using a weight vector to describe the system across chromosomes.

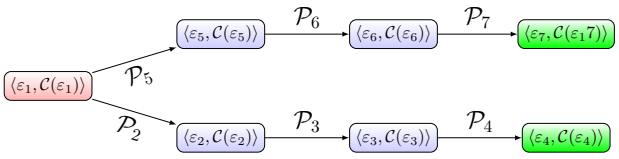

Fig. 4. A Configuration of the Election Process at a slot of time.

(1) Define an adequate *chromosome* which the weight vectors.
(2) Select a large set of chromosomes, *population* takes into account all weight vectors collected from the different components.
(3) The reproduction operators include selection, crossover, and mutation, which are all used in the process of enhancing the trained models. *selection* is applied 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 of all edge nodes models is more than 99%.

Algorithm 1 Federated Genetic Algorithm (FedGA), The **K** clients are indexed by **k**; **B** is the local mini-batch size, $\mathcal{D}_k$is the dataset available to client **k**, $\mathcal{D}_t$ is the dataset used for the test which is available on the aggregator, $W_{\mathbf{B}}$ is the vector of base layers (classification layers), $W_{\mathbf{P}}$ is the vector of personalized layers (features extractor layers), **E** is the number of local epochs, and η is the learning rate.

Procedure FedGA ▷ Run on the server.
$W_{\mathbf{B}}^0 \leftarrow Rand()$; ▷ Initialize the weights randomly.
for $\mathtt{t} = 1, 2, 3, \ldots$ **do** ▷ Round **t**.
  for $\mathtt{k} \in \mathtt{K}$ **do** ▷ To select a client **k**.
    $W_{\mathbf{B},\mathbf{k}}^{t+1} \leftarrow \mathbf{k}.Update(\mathrm{k}, \mathrm{W}_{;bfB,k}^{t})$; ▷ In Parallel call the function Update.
  end for
  $W_{\mathbf{B}}^{t+1} = GA(\mathcal{D}_t, W_{\mathbf{B}}^{t})$; ▷ Only base layers are aggregated
end for
End procedure FedGA
Procedure Update(k, w_B^t) ▷ Run on the client k.
$\beta \leftarrow$ (Split $\mathcal{D}_k$into mini-batches of size B;)
for **i** from 1 to **E** **do** ▷ **i** is an epoch.
  for **b** $\in \beta$ **do** ▷ **b** is a batch.
    $w_{\mathbf{B}}, w_{\mathbf{P}} \leftarrow w - \eta \Delta \mathcal{L}(w_{\mathbf{B}}, w_{\mathbf{P}}, b)$; ▷ base layers are updated and trained and personnalized layers are trained
  end for
end for
`return` **t** ▷ **t** will be stored in the Server.
End procedure

E. Broadcasting

After aggregation, `FedGA-ICPS` passes model base layer weights to learning phase components. In the training phase, a high-accuracy, low-loss model is created. Homomorphic encryption is used during broadcast phase. Distributed homomorphic cryptosystem (DHC) implements homomorphic operations utilizing secure multiparty computing (SMC).

IV. Experimental results

We evaluated the performance of our `FedGA-ICPS` framework using the Fashion MNIST dataset [14], it is a large collection of handwritten digits that is often used for the purpose of training a variety of image processing systems. We employed this dataset in order to test the effectiveness of our project framework. We divide it into four distinct subsets to better understand it. Each user has a unique component that differs from that of other users and is of uneven size (non-iid data). We carried out experiments using CNN to evaluate the effectiveness of federated learning when combined with genetic algorithms results, as opposed to FedAVG and FedPer, in comparison with the results obtained with FedAVG. Every user has their own convolutional layers, in addition to the classification layers that everyone else uses. To train the models , we have used a mini-batch SGD of size 1000, a ReLU activation function, and a dropout is employed to prevent overfitting. PyTorch was used throughout the development of the models for our various implementations. Using FedAVG as an aggregation algorithm in the case where the models are homogeneous, the results presented in Figure 6 show that there is a perturbation of the accuracy for each customer due to data heterogeneity and dataset size variance. This is shown by the fact that the FedAVG aggregation algorithm was used.

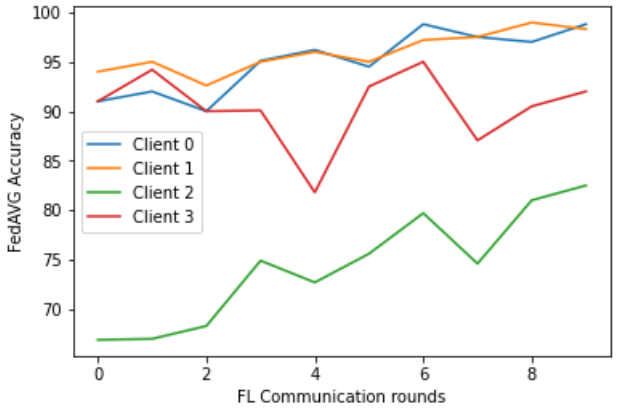

Fig. 5. FedAVG Aggregation Algorithm.

The FedPer model divides the model into two layers: the base layer and the personalized layer. Only the base layers are aggregated by the federated server in order to create a personalized layer, and only the base layers are communicated back to the server to be personalized. 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 99% as shown in Figure 7.

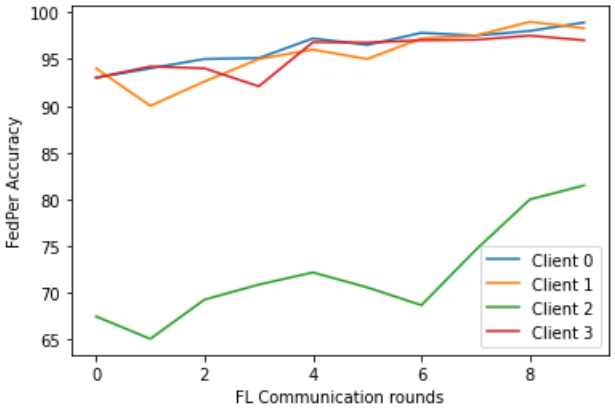

Fig. 6. FedPer Aggregation Algorithm.

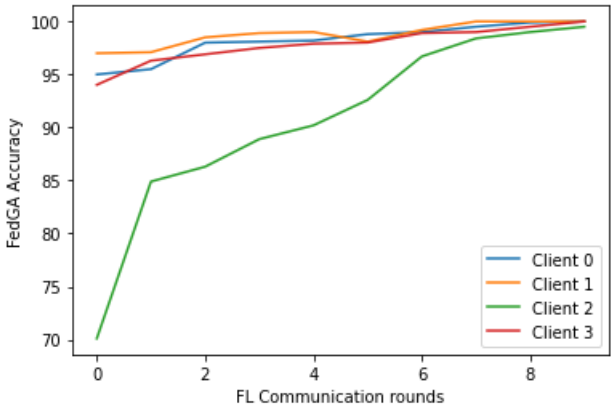

Fig. 7. FedGA Aggregation Algorithm.

V. Conclusion

In this research, we have provided a first step toward a complete approach for strengthening performance analysis in order to construct a more robust `ICPS` . This step was one of the goals that we set out to accomplish. A system is defined as a collection of items, each of which has its own structure and behavior for the goal of carrying out a given function, according to the framework that was designed for `FedGA-ICPS` . `FedGA-ICPS` plans to employ federated learning and genetic algorithms to aid restricted devices in locally embedding their decision models in order to speed up the analysis and learning processes. This will allow the project to do these tasks more quickly. By using a well-known standard, we were able to provide evidence that the proposed framework is effective. In the future, we plan to expand the capabilities of the framework by (1) incorporating additional machine learning techniques and automatically selecting the best agent, primarily through reinforcement learning; (2) decentralizing the system through the implementation of a blockchain architecture; and (3) evaluating the framework on use cases and benchmarks that are more complex.

References

[1] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv e-prints*, pp. arXiv–1602, 2016.

[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] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.