Fedga-meta: Federated learning framework using genetic algorithms and meta-learning for aggregation in industrial cyber-physical system

Badra Souhila Guendouzi, Samir Ouchani, Hiba Al-Assaad, Madeleine Jradi

FedGA-Meta- Federated Learning Framework using Genetic Algorithms and Meta-Learning for Aggregation in Industrial Cyber-Physical Systems

Souhila Badra GUENDOUZI
LINEACT CESI
Lyon, France
bguendouzi@cesi.fr

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

Hiba EL ASSAAD
LINEACT CESI
Toulouse, France
halassaad@cesi.fr

Madeleine EL ZAHER
LINEACT CESI
Toulouse, France
melzaher@cesi.fr

***Abstract*—In Industry 4.0, factories encounter significant challenges in making informed decisions to maintain or enhance their industry standing. By utilizing machine learning (ML), they can improve product quality, decrease production downtime, and boost operational efficiency. However, acquiring datasets with sufficient variation and diversity to train a robust neural network centrally is a challenge within the industrial sector. Consequently, federated learning (FL) offers a decentralized approach that safeguards data privacy, enabling smart infrastructures to train collaborative models locally and independently while retaining local data. In this paper, we present `FedGA-Meta` framework, which combines FL, meta-learning, and domain adaptation to enhance model performance and generalizability, particularly when training across distributed factories with varying network and data conditions. The results obtained demonstrate the effectiveness and efficiency of our `FedGA-Meta` framework.**

***Index Terms*—Industrial Cyber-physical systems, Federated Learning, Domain Adaptation, Meta-Learning, Data heterogeneity, Decision-making**

I. Introduction

With the abundance of data available today, AI techniques are being applied across various sectors. However, data confidentiality and user privacy constraints present a significant challenge in developing robust decision-making mechanisms, especially in industrial environments where safety and quality of service are crucial. In Industrial CPS (ICPS), components are connected in a centralized architecture with global decision-making based on collected data. FL [1] has emerged as a promising solution, enabling distributed client devices to collaboratively train a shared prediction model while keeping training data local. FL has gained considerable interest in the industry due to its advantages over traditional ML approaches. Further, it has been adopted by several industry giants, including Google, Apple, and Microsoft. Google, for example, used FL to improve the accuracy of its keyboard app, Gboard, by training the model on users' devices [2]. Similarly, Apple used FL to improve the accuracy of its QuickType keyboard, and Microsoft used it to train a model that predicts the likelihood of a Windows machine being infected with malware [3].

Despite the benefits of FL, training in heterogeneous and potentially complex networks presents significant challenges, particularly during the aggregation step, which must be addressed to fully realize its potential. These challenges encompass privacy preservation, limited communication, handling data and model heterogeneity, and reducing algorithm convergence complexity, among others [4].

The main goal of this work is to overcome the limitations of `FedGA-ICPS` [5] by integrating Meta-Learning and Domain Adaptation, which are ML techniques and aim to enhance model performance and generalizability while minimizing both communication and computation overhead in FL process. By incorporating these techniques, we strive to develop a more efficient and effective **`FedGA-Meta`** framework that enhances the aggregation process of FL and deploys it for ICPS. The main contributions of this paper are as follows.

- Surveying the main contributions related to the FL aggregation approaches and applications.
- Developing a framework to enhance the performance analysis and the decision support to `ICPS`.
- Modeling formally `ICPS` components and compositions.
- Proposing a federated solution to enhance the collaborative learning phases between `ICPS` components through genetic algorithms, meta-learning and domain adaptation techniques to reduce the domain shift problem.
- Studying the convergence and the effectiveness and efficiency of **`FedGA-Meta`**.
- Comparing **`FedGA-Meta`** within the existing solutions and validating it on benchmarks.

The next section surveys the contributions related to FL aggregation algorithms. Section III develops our proposed framework, **`FedGA-Meta`**, that is validated in Section IV. setImmediate0.45404126284403519setImmediate0.45404126284403519 Section V concludes the paper and give hints on our **`FedGA-Meta`** related perspectives.

II. Litterature

To collaborate learning and make enhancement in decision-making, FL works by having a server that collects local model weights for each communication round t, aggregates them to create a global neural network model with global weights w^t, and then broadcasts the latter to all participants to make local updates.

McMahan et al. [1] introduced "Federated Averaging" or **FedAVG** with the following steps: global model weights are initialized and broadcasted to clients, who then train their local models. Some clients send their progress to the central server for aggregation, which averages the local model weights to create a stronger aggregated model. This process is iteratively repeated in subsequent FL rounds involving all participants. The **FedPer** algorithm, proposed by Arivazhagan et al. [6] supposes that each local model consists of (1) base layers that are similar and shared by all participants, and (2) personalized layers that are unique to each participant. Thus, the participants communicate to the server, for the aggregation, the base layers instead of the totality of the model while retaining the other layers.

FedProx is an FL framework, proposed by Li et al. [4]. It is a generalization of **FedAVG** with some modifications to address the heterogeneity of data and systems (systems and statistical heterogeneity). Mainly, the participants optimize the loss function with a proximal regularization term to penalize the wide divergence between them. Wang et al. [7] proposed Federated Matched Averaging **(FedMA)**, which is a new layers-wise FL algorithm for modern CNNs and LSTMs. Layers are independently trained and communicated to the server. **FedMA** requires FL communication rounds equal to the number of layers in local models, that are similar for all participants. Briggs et al. [8] proposed FL with hierarchical clustering of local models to improve learning on non-IID data. They group similar clients' local models in order to mitigate their divergence and execute aggregation for each cluster of local models.

Hu et al.[9] proposed **MHAT**, a heterogeneous aggregation training for FL that addresses heterogeneous model architectures and client-server communication costs by focusing on the model outputs rather than its parameters. Xie et al. [10] proposed an asynchronous FL aggregation (**FedAsyn**) method that builds a global model for any incoming local model to improve the flexibility and scalability of FL. Ek et al.[11] proposed **FedDist** FL aggregation algorithm for Human Activity Recognition (HAR) that can change the architecture of their models by finding variations between particular neurons among the clients to mitigate their divergence in case of heterogeneous and non-IID datasets. **FedGA** Guendouzi et al. [5] is an aggregation algorithm with genetic algorithms to solve the problem of data and model heterogeneity. With **FedGA**, all participants upload only the base layer weights to the central server. Then, the latter calculates the new weights by calling the genetic algorithm, where a weight vector is used to be as a chromosome.

Assuming a centralized public dataset (source domain D_S) on the aggregator server for constructing the global model, with private datasets from multiple collaborators as target domains $(D_T^1, ..., D_T^N)$, `FedGA-ICPS` and other FL aggregation algorithms may face challenges due to domain shifts. Also, enhancing only part of the neural network models isn't ideal for decision-making, and deploying `FedGA-ICPS` with many collaborators can increase time complexity because of the increase number of collaborators that increase it complexity. The goal of **`FedGA-Meta`** is to overcome these limitations by extending `FedGA-ICPS` formalism and integrating meta-learning, domain adaptation, and FL. This unified framework aims to enable efficient learning, adaptation to varying data distributions using domain adaptation techniques, generalize learning to make **`FedGA-Meta`** scalable using meta-learning, and enhance the performance and applicability of models.

III. **`FedGA-Meta`** Framework

In this section, we introduce first the global overview of **`FedGA-Meta`** then we detail it according to its workflow. As shown in Fig. III, **`FedGA-Meta`** considers the architecture of an Industrial CPS with layers: *Edge*, *Fog*, and *Cloud*. It combines edge collaborators $\varepsilon_{type=e}$ with similar feature spaces and associates them to a specific fog $\varepsilon_{type=f}$, establishing a hierarchical representation. Also, it performs the aggregation at two different levels: *Fog* and *Cloud*. Relying on aggregation at two levels, **`FedGA-Meta`** targets to 1) preserve the personalization of data and neural network models, 2) address scalability and generalizability by employing meta-learning and domain adaptation techniques and 4) reduce the communication overhead between the collaborators and the central server. As depicted in Fig. III, **`FedGA-Meta`** workflow is a 7-steps framework executed as follows.

- **Step 1:** The global model, initialized or updated by the cloud server, is shared with fog servers in the hierarchical architecture, giving each fog server access to the latest global model as a starting point.
- **Step 2:** Fog servers fine-tune the global model and share the updated version with their edge collaborators.
- **Step 3:** Edge collaborators independently update their models with deep learning techniques tailored to their data and share their enhanced models with the fog servers in **Step 4**, which aggregate the models within the same feature space.
- **Step 5:** Fog servers receive and store local models, execute the FedGA aggregation algorithm using subdatasets from the connected edge collaborators, and perform the inner-update of the MAML [12], a meta-learning algorithm, allowing for efficient combination of local models and adaptation of the global model.
- **Step 6:** Fog servers calculate the loss associated with their updated models and share this metric with the cloud server, which uses this information to refine the global model and assess the framework's performance.
- **Step 7:** The cloud server executes the outer-update of the MAML algorithm [12], enabling the model to learn from a variety of tasks, with each reflecting the distinct data distribution associated with a separate fog layer in the hierarchical architecture. This approach effectively leverages the collective knowledge available across multiple fog layers.

In the following, we detail **`FedGA-Meta`** steps including the formalization of `ICPS`, the learning phase, and the aggregation process.

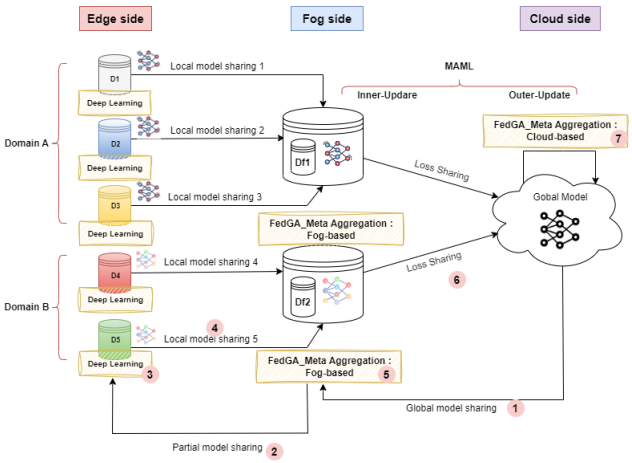

Fig. 1. An overview of **FedGA-Meta** Framework .

A. Industrial CPS Formalism

FedGA-Meta considers a system $\mathcal{S} = \langle \mathcal{E}, \mathcal{D}, \mathcal{N}, \mathcal{T}, \mathcal{CT} \rangle$ as a composition of a set of entities $\mathcal{E}$ that manipulate a specific data $\mathcal{D}$. They interact and interleave through a network of physical and logical channels ($\mathcal{N}$) to accomplish a precise task ($\mathcal{T}$) in a given context $\mathcal{CT}$.

1) Modeling the entities: An entity $\varepsilon \in \mathcal{E}$ can be an IIoT, an edge node, a fog node, or a cloud server that executes specific actions or collaborates with other entities to perform global tasks. To evaluate the guard related to an action, the entity ε run its associated machine learning model $n \in N_m$ that evaluates the variables of the specified guard. To enhance the decision-making of an entity ε, `FedGA-ICPS` develops techniques to help entities update periodically their associated ns. ε is the main entities describing `ICPS`, and it is defined by $\langle id, type, attr, Actuator, \Sigma, Beh \rangle$, where:

- id is a finite set of tags $id_1, \ldots, id_i, \ldots \in id$ identifying the entities.
- $type : id \rightarrow \{i, e, f, c\}$ returns the type of the entity; that can industrial IoT (i), edge (e), fog (f), or cloud (c).
- $attr : id \rightarrow 2^T$ returns the attributes of an entity, such that $T = \{p, l\}$ where p and l stand respectively for physical characteristics (memory, processing power, battery, etc) and a logic expression that specifies the performance of an entity.
- $Actuator$ specifies the status of an entity by evaluating its attributes,
- $\Sigma : id \rightarrow 2^A$ is a partial function that returns a finite subset of atomic actions. It depends on the type of entity ε_i to be executed by the latter, and initially, $A = \{$`Start`, `Send`, `Receive`, `Update`, `Predict`, `Train`, `Aggregate`, `Terminate`$\}$,
- $Beh : id \times \Sigma \rightarrow \mathcal{L}$ returns the behavior of an entity written in the language $\mathcal{L}$, and the syntax of $\mathcal{L}$ is given by: $B ::= \alpha \mid B \cdot B \mid B +_g B$, where $\alpha \in \Sigma$. The operator " $\cdot$ " composes sequentially the actions and $+_g$ is , by the functionality `Predict` . When $g^{\Delta} = \top$ the guarded decision become a non-deterministic choice.

2) Modeling the data: The data $\mathcal{D}$ describes the different data manipulated by the entities, exchanged and shared between them or broadcasted in the network. It is defined by a tuple $\langle \mathcal{E}, D_s, att_{D_s}, N_m, att_{N_m}, Mea, Ass \rangle$, where:

- D_s is a finite set of dynamic datasets (with element d_1, …, d_i, …, $d_{|D_s|}$), where $d_p = \{(x_k, y_k)\}_{k=1}^{|d_p|}$ is a public dataset shared between entities, and x_k and y_k are k-th sample and label respectively in d_p.
- $att_{D_s} : D_s \rightarrow 2^{\{type, domain, size, distribution, \mathcal{F}\}}$ returns the characteristics of a dataset $d \in D_s$, where they represent data type (e.g. images), domain of application (e.g. aeronautical equipment maintenance), dataset size, samples distribution (e.g. number of samples per class) and features representation of the dataset respectively.
- N_m is a finite set of neural network models (with element n_1, …, n_i, …,$n_{|N_m|}$.), where n_ϵ is the global model of the `ICPS`.
- $att_{N_m} : N_m \rightarrow 2^{\{type, task, w\}}$ returns the structure of a neural network model n, where type, task, and w represent its architecture (e.g. convolutional neural network), its output (e.g. object detection) and weights vectors respectively (e.g. n_ϵ is represented by the weights parameters w_ϵ).
  Ass: $\mathrm{E} \times 2^{\mathcal{F}} \times \mathcal{T} \rightarrow D_s \times N_m$ returns the architecture of a neural network n and its associated local dataset d.
- $Mea : D_s \times N_m \rightarrow 2^{\{accuracy, loss, F1-score\}}$ returns the evaluation measures of a neural network model $n \in N_m$ trained by a dataset $d \in D_s$.

3) 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. 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 $\mathcal{N} = \langle \mathcal{E}, \mathtt{Chan}, \mathtt{Prot}, \mathtt{Rel} \rangle$, where:

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

4) 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 $\langle \mathcal{G}, \preceq \rangle$, where:

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

5) 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

Initially, an entity $\varepsilon_{(id=i \wedge type=e)}$ provides its neural network model n_i and private dataset d_i. The goal of this process is to solve a DL problem locally (e.g. classification) by training n_i with d_i and getting by the end the evaluation metrics that need to be optimized. For example, using cross-entropy as the loss metric of a classification problem needs to be minimized as much as possible by applying an optimization algorithm, such as a stochastic gradient descent algorithm. The edge node $\varepsilon_{(id=i \wedge type=e)}$ can fine-tune a neural network model n_j from $\varepsilon_{(id=j \wedge type=f)}$ after the aggregation step. Algorithm 1 run locally on $\varepsilon_{(id=i \wedge type=e)}$ to fine-tune its initial model.

Algorithm 1 Local Update

1: **Function EdgeUpdate** ▷ Run on edge entity $\varepsilon_{(id=i \wedge type=e)} \in \mathcal{E}$.
2: **Input**
3: n_j partial model of $\varepsilon_{id=j \wedge type=f}$
4: B batch size
5: E local epochs
6: η learning rate
7: **Output**
8: w_i ▷ layers' weights of n_i.
9: $\beta \leftarrow$ (Split d_i into mini-batches of size B)
10: **for** each local epoch **i** from 1 to **E** **do**
11: **for** batch **b** $\in \beta$ **do**
12: $w_i \leftarrow w_i - \eta \Delta_{w_i} \mathcal{L}(n_{w_i}, b)$ ▷ Local learning.
13: **end for**
14: **end for**
15: **End Function EdgeUpdate**

C. Aggregation

In order to preserve the personalization of private data and neural network models as well as reduce the communication overhead between cloud and edge entities, **`FedGA-Meta`** divides the aggregation process into two levels, fog-based aggregation and cloud-based aggregation. The details of these are mentioned as follows:

1) Fog-based Aggregation: A fog entity $\varepsilon_{(id=j \wedge type=f)}$ groups a set of edge entities $\varepsilon_{(type=e)} \in \mathcal{E}$ that have the same feature spaces and share the same neural network architecture. Further, it provides a secure dataset d_j, that comprises a variety of distinct subdatasets derived from the connected edge collaborators, and a partial model n_j. When the aggregation process is started, each $\varepsilon_{type=f}$ initializes n_j and broadcasts it to $\varepsilon_{(type=e)} \in \mathcal{E}$. These latter train and update (Section III-B) their neural network models and share them with $\varepsilon_{type=f}$ for partial aggregation by applying **FedGA** [5]. The output is then a new n_j, presented by:

$$n_j = GA(n_j, d_j, W) \quad where \quad W = \cup_{i=1}^{E} w_i \tag{1}$$

As a result, each $\varepsilon_{type=f} \in \mathcal{E}$ will have a distinct n_j architecture. More precisely, each architecture has different personalized layers (e.g. feature extractors layers in CNN case) but it shares the same decision-making architecture (e.g. classification layers in CNN case). Afterward, n_j will be trained using the inner-update of MAML [12] algorithm.

Algorithm 2 `FedGA-Meta` Aggregation Step1

1: **Function FogAggregation** ▷ Run on cloud entity $\varepsilon_{(id=j \wedge type=f)} \in \mathcal{E}$
2: **Input**
3: w_ϵ Layer's weights of n_ϵ
4: $\mathsf{E} \in \mathcal{E}$ Set of associated edge entities
5: **Output**
6: $\mathcal{L}_Q^{\pi_j}(n_{w_j})$ Loss
7: $W = \phi$ ▷ Initialize an empty population of weight's vectors
8: **for** each edge entity $e \in$ E **do**
9: $W \leftarrow W \bigcup e.ClientUpdate(w_\epsilon)$ ▷ Get local layer's weights from each $\varepsilon_{type=f} \in \mathcal{E}$ and update the population
10: **end for**
11: $w_j = GA(n_j, d_j, W)$ ▷ Execute GA to get new weights
12: $w_j = w_j - \alpha \Delta_{w_j} \mathcal{L}_S^{\pi_j}(n_{w_j})$ ▷ Execute inner-update of MAML
13: `return` $\mathcal{L}_Q^{\pi_j}(n_{w_j})$
14: **End Function FogAggregation**

Each $\varepsilon_{(id=j \wedge type=f)}$ provides a meta-learning task π_j, where d_j is split into *support* set $d_S = \{(x_k, y_k)\}_{k=1}^{|d_S|}$ and a *query* set $d_Q = \{(x'_k, y'_k)\}_{k=1}^{|d_Q|}$. Therefore, n_i is trained using d_S in order to get a new weight's vector w_j. The inner-update of n_j is given by 2 as follows.

$$w_j = w_j - \alpha \Delta_{w_j} \mathcal{L}_S^{\pi_j}(n_{w_j}) \tag{2}$$

After that, w_j is split into $wj, base$ and wj, per, which represent respectively the weights of base and the personalized layers related ti n_j. In addition, $\varepsilon_{type=f \wedge id=j}$ evaluates n_j with d_Q and calculates the loss functions by freezing the personalized layers. Finally, the losses (the evaluations of the loss functions) $\mathcal{L}_Q^{\pi_j}(n_{w_j})$ will be shared with the aggregator for the second step of MAML as presented in Algorithm 2.

2) Cloud-based Aggregation: The cloud entity $\varepsilon_{type=c}$ receives $\mathcal{L}_Q^{\pi_j}(n_{w_j})$ from each $\varepsilon_{type=f}$ and applies the outer-update of MAML algorithm to get the global model n_ϵ by applying Eq. 3.

$$w_\epsilon = w_\epsilon - \frac{\gamma}{J} \Delta_{w_\epsilon} \sum_{j=1}^{|J|} \mathcal{L}_Q^{\pi_j}(n_{w_j}), \; where \; J = \cup \, \varepsilon_{type=f} \tag{3}$$

At the end of this step, $\varepsilon_{type=c}$ broadcasts w_ϵ to all $\varepsilon_{type=f} \in \mathcal{E}$ for another FL round, until achieving the final round. Algorithm 3 stepwise the phase of cloud-based aggregation.

IV. Experiments

In this section, we evaluate the effectiveness and efficiency of **FedGA-Meta** on different federated real datasets and models. Compared to the traditional FL approaches, we show that **FedGA-Meta** can provide faster convergence, higher accuracy, and also reduce the system overhead.

A. Experimental setup

FedGA-Meta has been implemented using PyTorch ML framework [13] with other neural network libraries, such as PyGAD [14] that helps to train neural network models using genetic algorithms. We use digit recognition as a real use case of our experiments, which is an image classification problem. Therefore, we selected MNIST [15], USPS [16], SVHN [17], MNISTM [18], and EMNIST [19] as benchmark datasets.

Our system consists of five $\varepsilon_{type=f}$ (5 fogs), where each groups four $\varepsilon_{type=e}$ (20 edges in total). We assign for each fog entity a different dataset (MNIST, SVHN, MNISTM or EMNIST) that is distributed unequally over its edge entities. Further, **FedGA-Meta** generates five heterogeneous CNN models that share the same decision-making out layers. To make it heterogeneous, it assigns a model to each fog entity. Then, the models are trained using mini-batches SGD, ReLU activation functions, and dropout functions to deal with overfitting. Also, they consist of two fully connected layers as base layers. We compare **FedGA-Meta** with **FedPer** and `FedGA-ICPS` frameworks, and also without considering FL paradigm (**NoFL**).

B. Experimental results

In order to assess the impact of the computing capacity on **FedGA-Meta** entities, we evaluate the average accuracy of their models by reducing the number of training epochs. We set three experiments: the number of local epochs equals to **1**, **5**, and **20**. For each experiment, we ran the four algorithms for 20 FL communication rounds. As it is shown in Figure 2, **FedGA-Meta** provides good results compared to other approaches. When the local epoch is equal to **1**, **FedGA-Meta** achieved an average accuracy of 88.28% compared with NoFL, FedGA, and FedPer that achieved an average accuracy of 55.09%, 82.45% and 84.76% respectively. We conclude that **FedGA-Meta** does not require entities possessing high computing capacity to participate in the FL process. So, we have only to increase the number of FL communication rounds to get the higher accuracy for all entities models $\mathcal{E}$.

Algorithm 3 **FedGA-Meta** **Aggregation Step1**

Function GlobalAggregation ▷ Run on cloud entity $\varepsilon_{(id=i \wedge type=c)} \in \mathcal{E}$
Input
  n_ϵ Global model
  J A set of fog entities $J \subset \mathcal{E}$
  γ Meta-learning hyperparameter
  F FL rounds
Output
  w_ϵ layers' weights of n_ϵ
$G = \phi$ ▷ Initialize an empty set of losses
for each FL round **k** from 1 to **F** **do**
  for each fog entity $j \in$ **J** **do**
    $G \leftarrow G \bigcup j.FogAggregation(w_\epsilon)$ ▷ Get losses from each $\varepsilon_{type=f} \in \mathcal{E}$
  end for
  $w_\epsilon \leftarrow w_\epsilon - \frac{\gamma}{J} \sum_{g \in G} g$ ▷ Outer-update of MAML

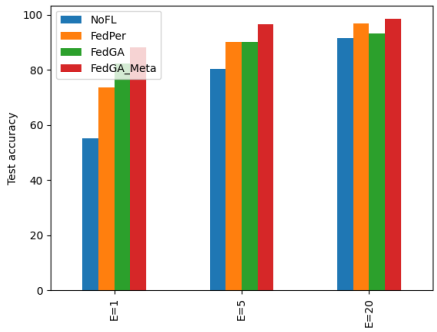

Fig. 2. The Performance of FL strategies on MNIST, USPS, SVHN, MNISTM, and EMNIST datasets for different local epochs E.

By taking into account the restricted processing power of edge entities, we set the default number of local epochs to only **1** for all experiments. Also, the number of FL communication rounds is fixed at 20. Table I summarizes the setup of our experiments and shows the final accuracy of each edge entity's local model. The results confirm that without implementing FL, the convergence will never be completed. In this case, implementing `FedGA-ICPS` or **FedPer** because of the domain shift problem for some of the local models. That is why proposing **FedGA-Meta** can tackle this limitation using MAML algorithm and gives higher accuracy.

Fog	Edge	NoFL	FedPer	FedGA	FedGA_Meta
$\varepsilon_{type=f}$	ε_1	49.46 %	86.19 %	87.07 %	93.68 %
	ε_2	47.54 %	87.94 %	92.20 %	97.38 %
	ε_3	3.22 %	90.23 %	90.38 %	98.33%
	ε_4	46.39 %	91.57%	93.54%	94.04%
$\varepsilon_{type=f}$	ε_5	79.59 %	88.37 %	89.30 %	97.69 %
	ε_6	95.49 %	83.65 %	79.93%	99.35 %
	ε_7	85.75 %	94.60 %	85.66 %	99.68 %
	ε_8	66.77 %	95.42 %	97.05 %	98.37 %
$\varepsilon_{type=f}$	ε_9	67.31 %	95.26 %	95.20 %	95.65%
	ε_{10}	76.53 %	96.06 %	96.35 %	96.90 %
	ε_{11}	74.10 %	97.27 %	96.58 %	97.52 %
	ε_{12}	57.79 %	93.94 %	93.27 %	91.67%
$\varepsilon_{type=f}$	ε_{13}	49.46 %	58.43 %	50.68 %	79.89 %
	ε_{14}	30.14 %	58.17 %	50.26 %	78.23%
	ε_{15}	0.252 %	72.10 %	69.83 %	80.02%
	ε_{16}	46.74 %	66.06 %	57.38 %	81.56 %
$\varepsilon_{type=f}$	ε_{17}	17.92%	84.03 %	74.70 %	95.21 %
	ε_{18}	11.50 %	74.42 %	69.02 %	94.82 %
	ε_{19}	54.13 %	90.00 %	76.97 %	99.90 %
	ε_{20}	0.31 %	90.14 %	88.45 %	92.21%

TABLE I
The accuracies of Final models at each edge entity.

V. Conclusion

In this work, we studied some of the existed FL aggregation methods. Also, we have given a first step toward a comprehensive methodology for enhancing performance analysis in order to create a more robust ICPS. We have designed and implemented a framework, called **FedGA-Meta**, based on FL dedicated to ICPS, focusing on the aggregation method that it is implemented with genetic algorithms, meta-learning and domain adaptation based approaches. The developed **FedGA-Meta** 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, in the experiment part, we demonstrated the efficacy of the suggested framework using renowned benchmarks with different scenarios. As a future work, we intend to expand the framework in many directions, mainly:

1) Incorporating additional ML techniques and selecting automatically the best agent for the aggregation process. This objective can be achieved and tested first by applying reinforcement learning.
2) The case when a model is weak and no pre-trained model already exists, a new mechanism to handle this issue could be considered by relying on data fusion, mimicking different nodes, and generating data using stochastic models.
3) Covering more ICPS architecture by decentralizing the system using blockchain schemes.
4) 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] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," arXiv preprint arXiv:1811.03604, 2018.

[3] D. Dimitriadis, M. H. Garcia, D. M. Diaz, A. Manoel, and R. Sim, "Flute: A scalable, extensible framework for high-performance federated learning simulations," arXiv preprint arXiv:2203.13789, 2022.

[4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," Proceedings of Machine Learning and Systems, vol. 2, pp. 429–450, 2020.

[5] S. B. Guendouzi, S. Ouchani, and M. Malki, "Enhancing the aggregation of the federated learning for the industrial cyber physical systems," in 2022 IEEE International Conference on Cyber Security and Resilience (CSR). IEEE, 2022, pp. 197–202.

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

[7] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," arXiv preprint arXiv:2002.06440, 2020.

[8] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020, pp. 1–9.

[9] L. Hu, H. Yan, L. Li, Z. Pan, X. Liu, and Z. Zhang, "Mhat: an efficient model-heterogenous aggregation training scheme for federated learning," Information Sciences, vol. 560, pp. 493–503, 2021.

[10] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," arXiv preprint arXiv:1903.03934, 2019.

[11] S. Ek, F. Portet, P. Lalanda, and G. Vega, "A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison," in 2021 IEEE International Conference on Pervasive Computing and Communications (PerCom) (PerCom 2021), Kassel, Germany, Mar. 2021.

[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in International conference on machine learning. PMLR, 2017, pp. 1126–1135.

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32. Curran Associates, Inc., 2019, pp. 8024–8035.

[14] A. F. Gad, "Pygad: An intuitive genetic algorithm python library," arXiv preprint arXiv:2106.06158, 2021.

[15] 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.

[16] J. J. Hull, "A database for handwritten text recognition research," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 5, pp. 550–554, 1994.

[17] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011.

[18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," The journal of machine learning research, vol. 17, no. 1, pp. 2096–2030, 2016.

[19] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in 2017 international joint conference on neural networks (IJCNN). IEEE, 2017, pp. 2921–2926.