Genetic Algorithm Based Aggregation for Federated Learning in Industrial Cyber Physical Systems

Souhila Badra Guendouzi, Samir Ouchani, Mimoun Malki

▶ **To cite this version:**

Souhila Badra Guendouzi, Samir Ouchani, Mimoun Malki. Genetic Algorithm Based Aggregation for Federated Learning in Industrial Cyber Physical Systems. International Joint Conference 15th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2022) 13th International Conference on EUropean Transnational Education (ICEUTE 2022), May 2023, SALAMANCA (SPAIN), France. pp.12-21, ⟨10.1007/978-3-031-18409-3_2⟩. ⟨hal-04086923⟩

HAL Id: hal-04086923

https://hal.science/hal-04086923v1

Submitted on 12 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright - All rights reserved

Genetic Algorithm based Aggregation for Federated Learning in Industrial Cyber Physical Systems

Souhila Badra GUENDOUZI and Samir OUCHANI and Mimoun MALKI

Abstract During the last decade, Industrial Cyber-Physical Systems (ICPS) have attracted a significant amount of interest from industries as well as academic institutions. These kinds of systems have proved to be very complicated, and it may be a difficult task to get a handle on their architecture and make sure everything works properly. By putting up a framework for federated learning that we've dubbed FedGA-ICPS the purpose of this study is to address some of the difficulties that are associated with the performance and decision-making aids provided by ICPSṪo begin, we launch an ICPS modeling formalism with the goal of specifying the structure and behaviour of such systems. FedGA-ICPS then conducts an analysis of the performance of the industrial sensors based on the data supplied by the ICPS from the *industrial* sensors by putting forth locally integrated learning models. Following that, a genetic algorithm drives federated learning in order to quicken and enhance the aggregation process. In the end, transfer learning is used so that the learned parameters of the models may be distributed across a variety of limited entities. FedGA-ICPS has been implemented on MNIST, and the results have been rather significant.

1 Introduction

The Internet of Things (IoT) has emerged as one of the most essential technologies of the 21st century during the years. There are many different fields, such as the Internet of Industrial Things (IoIT) and the Internet of Medical Things (IoMT), which refer to the improvement of industrial processes and medical applications through the

Souhila Badra GUENDOUZI
ESI Engineering School, Sidi bel Abbès, Algeria, e-mail: b.guendouzi@esi-sba.dz

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

Mimoun MALKI
ESI Engineering School, Sidi bel Abbès, Algeria, e-mail: m.malki@esi-sba.dz

utilization of smart sensors, actuators, quick communication protocols, and effective cybersecurity mechanisms. In vast networks, intelligent devices create a significant quantity of data; hence, IoT frameworks demand methods that are both clever and resilient for analyzing massive volumes of data.

Precisely, deep learning (DL) approaches have achieved promising results in networks application owing to their intelligent learning and processing skills. These qualities have enabled these techniques to create positive outcomes. Historically, the application of AI in CPS has been done in a centralized manner. This means that there is a central server that trains the model by using all of the data from the end devices that are linked to it. The transmission of such a massive volume of data from these end devices to the cloud, however, leads to congestion on the bandwidth, delays in the processing of data, and perhaps a breach of privacy. Recently, this method of learning has been modified to be either distributed learning or federated learning, which brings together a collection of clients (end devices) that each have their own set of data that is kept private and is not shared with other clients. They coordinate their efforts with one another so that they may achieve the same degree of learning by exchanging information on the learning parameters.

Google first presented the concept of federated learning (FL), which is a decentralized and collaborative method of machine learning, in the year 2016. The goal of the FL process is to produce a model that is suitable for a certain application in its entirety. Choose the appropriate machine learning model to use and select devices candidates to participate in the learning process where each device initializes its embedded model parameters and trains the model until it converges. This is an example of a typical workflow for its Life cycle, which includes: choosing the appropriate machine learning model to use. Following a training session on the local level, each of the linked devices then uploads its model parameters to the centralized server via a secure communication method that we will go over in more detail later. Then, after the aggregator (for example, a central server) has received all of the local models, it will aggregate the parameters in order to update the new global and optimum model. Once again, the server will re-share the parameters of the global model, and the devices will update their own parameters based on the new information. This procedure is carried out in a loop until the whole of the training process has been completed.

In the literature, McMahan et al. [1] introduced a new aggregated model called FedAVG, which stands for federated average. In this model, the weights of the several local models are averaged by the server, which then provides new weights and, therefore, a new model.

Also, FedPer proposed by Arivazhagan et al. [2] similar 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

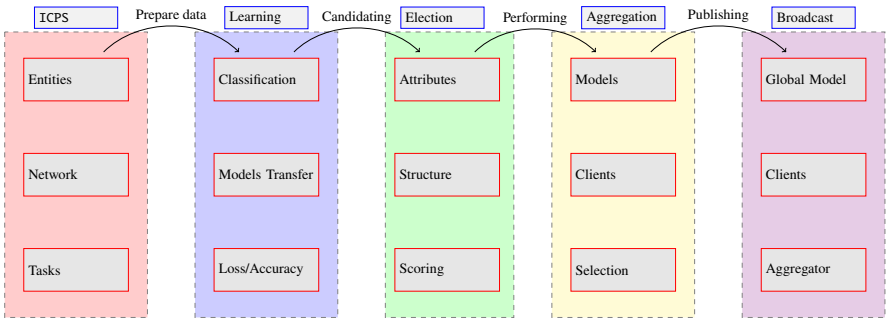

Fig. 1 The Interoperability and Integrity Validation and Evaluation.

deploy it for ICPS. As shown in Figure 1, FedGA-ICPS develops five stages: CPS (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 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 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.

In a nutshell, we present our main contributions in the present paper.

- 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 2 surveys contributions related to federated learning and its applications. Section 3 develops our

proposed framework, FedGA-ICPS that is validated in 4. Finally, Section 5 concludes the paper and give hints on our FedGA-ICPS related perspectives.

2 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 medical data collected from IoMT stored in a blockchain. The solution implements three agents: learning, indirect, and data management (DM). 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] used an asynchronous FL-based anomaly detection for resources constrained IoT devices using deep learning techniques. Their approach is established through: pre-initialization of the parameters, and deploy an asynchronous model training. Unfortunately, they did not study the reliability of the participating nodes. 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 issues of data isolation and model personalising in FL. Initially, the cloud server develops the global model using public datasets, distributes it to the clients using homomorphical encryption, then the clients retrain local models and upload them to the cloud server. Monotonously, the aggregator builds 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 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.

3 FedGA-ICPS Framework

This section follows FedGA-ICPS steps depicted in Fig. 1 by firstly presenting ICPS formalism.

3.1 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 ($\mathcal{T}$) in a specific context $\mathcal{CT}$. A system $\mathcal{S}$ is a tuple $\langle \mathcal{E}, \mathcal{N}, \mathcal{T}, \mathcal{CT} \rangle$.

3.1.1 Modeling the entities

An entity $\varepsilon \in \mathcal{E}$ can be an IIoT, an edge node, a fog node or a cloud server that 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 $\texttt{ml}_\varepsilon$ 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, ε is the main entities describing ICPS, and it is defined by the tuple $\langle id, attr, Actuator, , \Sigma, Beh \rangle$, 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,
- is a function that associate to an entity ε its ml.
- $\Sigma = \{$Send, Receive, Update, Predict, Train, Aggregate$\}$ is a finite set of atomic actions that depend on the type of entity ε_i and executed by the latter.
- $Beh: id \times \Sigma \rightarrow \mathcal{L}$ 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 Predict. When $g \triangleq \top$ the guarded decision become a non-deterministic choice.

3.1.2 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 $\mathcal{N} = \langle \mathcal{E}, \texttt{Chan}, \texttt{Prot}, \texttt{Rel} \rangle$, 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.

3.1.3 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 $\texttt{Goals}$.

3.1.4 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.

3.2 Learning

3.2.1 Classification

In the learning step, FedGA-ICPS relies on CNN, convolutional neural network, which is a class of deep neural network [9]. A given CNN is locally learned in a machine by considering only its proper data. The architecture of CNN consists of three layers: (1) convolution, (2) pooling, and (3) fully connected. The convolution process with several filters is capable of extracting features from the data set. Pooling is a technique for reducing the dimensionality of feature maps. The most popular pooling methods employed in CNN are maximum pooling and average pooling [10]. The fully-connected layer is in charge of categorizing the data based on the characteristics collected by the previous layers. While ReLu functions are often used to categorize inputs in convolutional and pooling layers, FC layers generally employ a softmax activation function to offer a probability from 0 to 1. Each edge client (manufacturing) has its own CNN model in our system, which does not have to be identical to other local models in terms of number of filters, layer architecture, function activation, etc., because it is dependent on its own dataset's structure. At training stage, Zhang and Sabuncu[11] argued that the most common way to train CNNs for classification problems is to couple stochastic gradient descent coupled with the Cross Entropy (CE).

3.2.2 Models Transfer

As the deep neural network needs a large amount of data and resources with computing power to accelerate learning to have good decision-making, the reuse of a pre-trained model on a related learning task is known as transfer learning (TL) that is considered as a solution to deal with these challenges. Two ways are possible to exploit transfer learning process: (1) Fine-tuning pre-trained-models, which uses a pre-trained model with a source dataset and retrain it with a target dataset. (1) Fine-tuning feature extractors, which freezes feature extractor layers and retrain only classification layers. In our study, we focus on Fine-tuning feature extractors. Whenever a new edge node joins the industrial network, the cloud server looks first for similar pre-trained models deployed on other similar nodes that were trained using a large dataset and achieved high accuracy. This then results in only retraining the base layers, rather than the whole network, since it transfers only the feature extractor layers to the target node.

3.3 Election

To elect the appropriate candidate for a federated learning, FedGA-ICPS looks for the most powerful component that can be a Fog or Edge node. As default, we consider the computing cloud server as the aggregator, then, depends on their capacities e.g 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.

3.4 Aggregation

Compared to FedAvg [12] and FedPers [2], with FedGA developed in FedGA-ICPS, all edge nodes upload only the encrypted base layer (classification layers) weights to the elected aggregator. Then, the latter calculates the new weights by calling the genetic algorithm (line 9) that runs as follows where a weight vector is used to represent the system throughout the chromosomes.

(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) 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 of all edge nodes models is more than 99%.

Algorithm 1 Federated Genetic Algorithm (FedGA), C is the fraction of clients selected randomly to participate in each communication round. 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), **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` **t** = 1, 2, 3, ... **do**
  $m \leftarrow max(C.K, 1)$;
  $S_t \leftarrow$ `(random set of` **m** `clients)`;
  for `each client` **k** $\in S_t$ **do**
    $W_{\mathbf{B},\mathbf{k}}^{t+1} \leftarrow ClientUpdate(\mathrm{k}, \mathrm{W}_{;bfB,k}^{t})$; ▷ In Parallel.
  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 ClientUpdate(k, w_B^t) ▷ Run on client **k**.
$\beta \leftarrow$ (Split $\mathcal{D}_k$into mini-batches of size B;)
for each local epoch **i** from 1 to **E** **do**
  for batch **b** $\in \beta$ **do**
    $w_{\mathbf{B}}, w_{\mathbf{P}} \leftarrow w - \eta\Delta\mathcal{L}(w_{\mathbf{B}}, w_{\mathbf{P}}, b)$; ▷ Only base layers are aggregated
  end for
end for
`return` **t** `to the Server`
End procedure ClientUpdate($k, w_{\mathbf{B}}$)

3.5 Broadcasting

After the aggregation phase, `FedGA-ICPS` transmits the weights of the model's base layers to participating components in the learning phase. Thus, a new model with high accuracy and low losses is generated in the training phase. Thus, broadcast phase is provided with Homomorphic encryption represented.

4 Experimental results

We evaluated the performance of our `FedGA-ICPS` framework using the MNIST dataset [13]. 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 show that there is a perturbation of the accuracy for each customer due to data heterogeneity and dataset size variance. 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. The results show that is a perturbation of the accuracy for each customer due to data heterogeneity and dataset size variance. The obtained results 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 2.

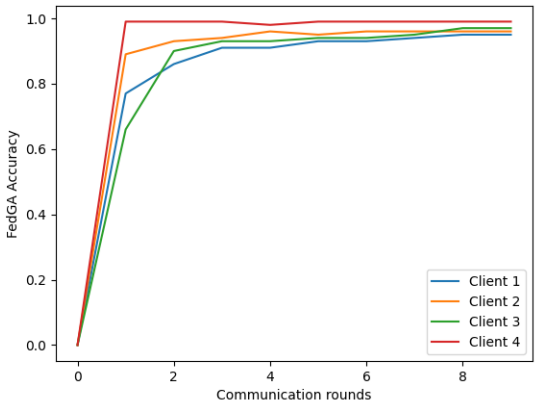

Fig. 2 FedGA Aggregation Algorithm.

5 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 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] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[3] Sannara Ek, François Portet, Philippe Lalanda, and German Vega. A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison. In *19th IEEE I. C. on Perv. Comp. and Comm. PerCom*, 2021.

[4] Dawid Połap, Gautam Srivastava, and Keping Yu. Agent architecture of an intelligent medical system based on federated learning and blockchain technology. *Journal of Information Security and Applications*, 58:102748, 2021.

[5] Pu Tian, Zheyi Chen, Wei Yu, and Weixian Liao. Towards asynchronous federated learning based threat detection: A dc-adam approach. *Computers & Security*, 108:102344, 2021.

[6] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 35(4):83–93, 2020.

[7] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. on Ind. Informatics*, 16(10):6532–6542, 2019.

[8] Chunyi Zhou, Anmin Fu, Shui Yu, Wei Yang, Huaqun Wang, and Yuqing Zhang. Privacy-preserving federated learning in fog computing. *IEEE Internet of Things Journal*, 7(11):10782–10793, 2020.

[9] Chapter 4 - multi-category classification problem. In Shih-Chia Huang and Trung-Hieu Le, editors, *Principles and Labs for Deep Learning*, pages 81–116. Academic Press, 2021.

[10] Wenbo Zhu, Yan Ma, Yizhong Zhou, Michael Benton, and Jose 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)*, volume 44, pages 2245–2250. Elsevier, 2018.

[11] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[12] Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging. *arXiv preprint arXiv:2104.11375*, 2021.

[13] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.