Particle Swarm Optimized Federated Learning For Industrial IoT and Smart City Services

Basheer Qolomany*, Kashif Ahmad†, Ala Al-Fuqaha†, and Junaid Qadir‡
*Department of Cyber Systems, College of Business & Technology, University of Nebraska at Kearney, Kearney, NE 68849 USA, qolomanyb@unk.edu
†Information & Computing Technology Division, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar, {kahmad, aalfuqaha}@hbku.edu.qa
§Department of Electrical Engineering, Information Technology University, Lahore, Pakistan, junaid.qadir@itu.edu.pk

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9322464

***Abstract*—Most of the research on Federated Learning (FL) has focused on analyzing global optimization, privacy, and communication, with limited attention focusing on analyzing the critical matter of performing efficient local training and inference at the edge devices. One of the main challenges for successful and efficient training and inference on edge devices is the careful selection of parameters to build local Machine Learning (ML) models. To this aim, we propose a Particle Swarm Optimization (PSO)-based technique to optimize the hyperparameter settings for the local ML models in an FL environment. We evaluate the performance of our proposed technique using two case studies. First, we consider smart city services, and use an experimental transportation dataset for traffic prediction as a proxy for this setting. Second, we consider Industrial IoT (IIoT) services, and use the real-time telemetry dataset to predict the probability that a machine will fail shortly due to component failures. Our experiments indicate that PSO provides an efficient approach for tuning the hyperparameters of deep Long short-term memory (LSTM) models when compared to the grid search method. Our experiments illustrate that the number of client-server communication rounds to explore the landscape of configurations to find the near-optimal parameters are greatly reduced (roughly by two orders of magnitude needing only 2%– 4% of the rounds compared to state of the art non-PSO-based approaches). We also demonstrate that utilizing the proposed PSO-based technique to find the near-optimal configurations for FL and centralized learning models does not adversely affect the accuracy of the models.**

***Index Terms*—Federated Learning, Deep Learning, Particle Swarm Optimization, Parameter Optimization, IoT, Smart Services**

I. Introduction

In recent years, the enormous growth in the processing and storage capabilities of end devices, and an ever-increasing concern about data privacy has led to a growing interest in Federated Learning (FL) [1]. FL, which builds a Machine Learning (ML) model across multiple decentralized edge devices, empowers multiple stakeholders (i.e., data owners) to benefit from each others' data by training a common and robust ML model collaboratively without exposing their data to other participants. The core concept of FL is based on training local models on the local resources (i.e., training samples) and then sharing the model's parameters with the other parties via a central server or in a peer-to-peer (P2P) manner depending on the FL topology to build a global model. Combining edge computing characteristics with data privacy, FL becomes an attractive proposition for a diverse set of applications—such as sentiment analysis, monitoring activities of mobile users, autonomous vehicles, and healthcare [2], [3], where data is distributed at multiple devices.

The feasibility of FL is highly dependent on the capabilities of edge devices to perform efficient local training and inference with the performance also depending on the appropiate tuning of the ML models and their parameters. For instance, the performance of deep neural networks (DNNs) depends on parameters, such as the number of hidden layers, the number of neurons in a layer, and the total number of epochs [4]. One of the main challenges in efficient training at the edge devices is the selection of the appropriate structure of the local ML algorithm and parameter settings.

The data privacy aspects and building a global model from distributed and heterogeneous data sources pose several challenges for FL in terms of optimization, data privacy, and efficient communication. In particular, existing works on FL assume fixed settings (i.e., architectures/structure and other parameters) for the ML models trained on the local dataset at the local hosts without exploring the optimal values for these parameters. Though there are several heuristic rules of thumb [4], [5], there is no theory yet to define the structure of ML models for a particular application. The rules of thumb, such as "*number of hidden layers in NNs should always be less than or equal to the number of inputs to the NNs* [5]" and "*the number of hidden layer neurons should be less than twice of the number of neurons in the input layer* [6]" do not always guarantee better results as they do not consider different factors, such as the number of training epochs, noise in the data, and the complexity of the objective function [4]. Moreover, the manual selection of these parameters using a trial and error method is cumbersome and requires an extensive experimental setup.

In this paper, we propose a Particle Swarm Optimization (PSO)-based parameter setting framework for the common parameters of the local models trained by edge devices on the local data in an FL environment. More specifically, we use PSO to optimize three different parameters of local models—namely, (i) the number of hidden layers (L), (ii) the number of neurons in each layer (N), and (iii) the number of epochs (E) representing the number of training passes each client makes over its local dataset in each round of the FL algorithm.

PSO and other heuristic techniques introduce randomness and depend on luck to approach an optimal solution. They are designed to solve a problem within a reasonable computation time and produce a solution that is good enough for solving the problem at hand. Since there is no guarantee that the solution found using heuristic techniques will be the optimal solution, the solution is called "sub-optimal." Our choice of PSO as an optimization method is motivated by its simplicity, easy implementation, robustness, and fast convergence. Such characteristics make it a preferred choice in diverse application domains [7], [8]. This work is inspired by our previous work [4], where we proposed a PSO-based optimization framework for parameter tuning of deep learning models. In this work, we use a similar concept to optimize a FL framework by exploring the potential solution space instead of assuming a fixed structure and parameters for the local ML models.

The main contributions of the work can be summarized as:

(i) We propose a PSO-driving federated learning framework, where PSO-based optimization has been employed to tune the common parameters of the local models trained on edge devices.

(ii) We evaluate and compare the proposed PSO-based parameter optimization approach with the grid search technique to find the best configurations for local models in an FL environment.

(iii) We also analyze the error rates of the best configurations found using the proposed PSO and grid search for FL and centralized learning models.

The rest of the paper is organized as follows: Section II provides an overview of the related work. Section III details the proposed PSO-based parameter optimization model while Section IV describes the dataset and its split between different nodes. Section V provides the details of the conducted experiments, experimental setup, and experimental results. Finally, Section VII provides concluding remarks.

II. Related Work

Several interesting optimization techniques have been proposed to deal with the challenges associated with learning a ML model in FL environment (such as the non-IID, unbalanced, and highly distributed nature of data) [9]–[11]. For instance, to deal with unbalanced and non-IID data, Mcmahan et al. [9] proposed a synchronous update scheme called FedAvg for the optimization of the global model, where a fraction of clients are selected among the fixed number of clients each with a fixed local dataset by the sever to share the current global model's parameters. The selected clients then perform computation based on global parameters and their local dataset and resend the updates to the server. The server then uses these updates to compute the global model by averaging local Stochastic Gradient Descent (SGD) updates and repeats the process. Though the FedAvg technique has proved itself to be effective in dealing with unbalanced and non-IID data, it does not guarantee convergence in the presence of heterogeneous data [11]. To deal with heterogeneous data in FL, Sahu et al. [11] proposed a modified version of FedAvg, namely FedProx, by adding a proximal term to the objective function of FedAvg, which helps to improve the stability of the method against heterogeneous data. On the other hand, in [12], a multi-task learning-based solution, namely MOCHA, has been proposed to deal with the statistical challenges associated with FL by learning separate but related models for each node, simultaneously. Several other interesting solutions have been proposed to ensure convergence in the presence of heterogeneous data using assumptions such as convexity and uniformly bounded gradients [10], [13].

There is a lot of research literature dealing with parameter optimization of ML models with different techniques being proposed. For instance, Tran et al. [14] analyzed the impact of parameters optimization on the performance of traditional classification algorithms. In this regard, several optimization techniques have also been employed for optimizing the parameters of Support Vectors Machines (SVMs) [15]–[17]. Several methods have also been proposed for the optimization of different parameters of Artificial Neural Networks (ANNs) [18]–[20]. Some of the methods aim at optimizing the initial weights of NNs. For instance, in [18], instead of the traditional Backpropagation (BP) strategy for Functional Link Neural Networks (FLNN), a Bee Colony (BC) optimization algorithm has been employed to set the weights of the NNs. BC along with PSO-based methods have also been employed in [19], where both methods are used independently and in combination to optimize the computational parameters (i.e., weights and biases) for a Multi-Layer Perceptron (MLP) neural network. Apart from the initial weights and biases, the performance and the feasibility of NNs in an application also depends on other parameters, such as the number of hidden layers, learning rate, and momentum coefficients. To this aim, several techniques have also been proposed to optimize these parameters. For instance, Mirjalili et al. [21] proposed a hybrid PSO and gravitational search algorithm to optimize the learning rate and momentum of NNs. Similarly, Francesco et al. [22] proposed psoCNN, an optimized CNN architecture for image classification where PSO is used to identify an optimized CNN architecture achieving better performance against state-of-the-art architectures. In [23], a linearly decreasing weight particle swarm optimization (LDWPSO) method is proposed for optimizing hyperparameters of a CNN architecture.

The literature depicts that most of the existing works focus on global optimization, privacy, and communication aspects to analyze the feasibility of FL. However, we believe that the feasibility of FL is also highly constrained by the capabilities of edge devices to perform efficient local training and inference. Despite its importance, no prior work has analyzed this aspect of FL. To the best of our knowledge, the paper is the first work to focus on this important yet overlooked aspect of FL.

III. PSO-Based Parameter Optimization Model

The PSO was originally developed in 1995 by Kennedy and Eberhart [24], and inspired by the behavior of animal groups, such as bird and fish swarms [25]. The PSO is an iterative optimization method that has extensive capabilities for global optimization in addition to easy implementation, scalability, robustness, and fast convergence. It employs simple mathematical operators and is computationally inexpensive in terms of both memory requirements and speed [4]. The PSO algorithm is composed of a group of particles called a swarm, each particle represents a possible solution to the problem. These particles in the swarm repeatedly communicate with each other. Each particle is associated with its position, velocity and fitness value that is determined by an optimization function. The position of each particle in the swarm is updated to move closer to the particle which has the best position. While the velocity value conveys the direction and distance of the particle's movement.

In the proposed work, the process starts by initializing a group of random particles for L, N, and E, then it searches for an optima by updating generations. In every iteration, each particle is updated by following the two "best" values. The first one is the best solution (accuracy) it has achieved so far. This value is called *pbest* (the particle best position). Another best value that is tracked by the particle swarm optimizer is the best value obtained so far by any particle in the population. This best value is a global best and called *gbest* (the global best position). Each particle updates its velocity and position by tracking the values of *pbest* and *gbest* in each iteration.

The velocity and position of each particle for the number of hidden layers are updated according to the Equations (1) and (2):

$$\begin{aligned} V_{L,i}^{t+1} = w.V_{L,i}^{t} + c_1.rand.(L_i^{best} - V_{L,i}^{t}) \\ +c_2.rand.(G^{Lbest} - V_{L,i}^{t}) \end{aligned} \tag{1}$$

Where V_L is the velocity of the number of hidden layers, L_i^{best} is the particle's best local value of the number of hidden layers, and G^{Lbest} is the best global value of the number of hidden layers. The position for the number of hidden layers of particle i is updated based on the particle's velocity V_L in Eq(2):

$$L_i^{t+1} = L_i^t + V_{L,i}^{t+1} \tag{2}$$

The velocity and position of each particle for the number of neurons in each layer layer are updated according to the Equations (3) and (4) as follows:

$$\begin{aligned} V_{N,i}^{t+1} = w.V_{N,i}^{t} + c_1.rand.(N_i^{best} - V_{N,i}^{t}) \\ +c_2.rand.(G^{Nbest} - V_{N,i}^{t}) \end{aligned} \tag{3}$$

Where V_N is the velocity of the number of neurons in each hidden layer, N_i^{best} is the particle's best local value of the number of neurons in each hidden layer, and G^{Nbest} is the best global value of the number of neurons in each hidden layer. The position for the number of neurons of particle i is updated based on the particle's velocity V_N in Eq. (4):

$$N_i^{t+1} = N_i^t + V_{N,i}^{t+1} \tag{4}$$

The velocity and position of each particle for the number of epochs are updated according to the Equations (5) and (6) as follows:

$$\begin{aligned} V_{E,i}^{t+1} = w.V_{E,i}^{t} + c_1.rand.(E_i^{best} - V_{E,i}^{t}) \\ +c_2.rand.(G^{Ebest} - V_{E,i}^{t}) \end{aligned} \tag{5}$$

Where V_E is the velocity of the number of epochs, E_i^{best} is the particle's best local value of the number of epochs, and G^{Ebest} is the best global value of the number of epochs. The position for the number of epochs of particle i is updated based on the particle's velocity V_E in Eq. (6):

$$E_i^{t+1} = E_i^t + V_{E,i}^{t+1} \tag{6}$$

In Equations (1), (3) and (5), c_1 and c_2 are the acceleration coefficients and represent the weights of approaching the $pbest$ and $gbest$ of a particle. w is the inertia coefficient as it helps the particles to move by inertia towards better positions. $rand$ is a uniform random value between 0 and 1. The parameters utilized in our experiments are listed in Table I.

TABLE I
THE PARAMETERS UTILIZED IN OUR EXPERIMENTS

Parameter	Value
Population size	5
Learning coefficient, c_1, c_2	uniformly distributed between [0,4]
Maximum number of iterations	10
Number of hidden layers	within the range [1, 5] increment by 1
Number of neurons in each layer	within the range [1, 200] step size varied
Number of epochs	within the range [1, 50] increment by 5
Number of clients	5
Number of clients-server communication rounds (CommRounds) for a configuration settings	15
Total Number of clients-server communication rounds	= CommRounds × the number of tried model configurations to get the best
Hidden layer velocity	•$MinLayerVelocity = -0.1 \times (MaxLayers - MinLayers)$ •$MaxLayerVelocity = +0.1 \times (MaxLayers - MinLayers)$
Neuron velocity	•$MinLayerVelocity = -0.1 \times (MaxNeurons - MinNeurons)$ •$MaxLayerVelocity = +0.1 \times (MaxNeurons - MinNeurons)$
Epoch velocity	•$MinLayerVelocity = -0.1 \times (MaxEpochs - MinEpochs)$ •$MaxLayerVelocity = +0.1 \times (MaxEpochs - MinEpochs)$

Algorithm 1 describes our proposed PSO-based parameter selection technique for deep LSTM models. The server starts by initializing random parameter configurations for the number of hidden layers, the number of neurons in each layer, and the number of epochs of deep LSTM models. The algorithm returns the near-optimal configuration. The algorithm is presented for two use-case scenarios (i.e., smart city services and the IIoT services proxy use cases), which will be fully explored in Section V.

IV. Datasets

In the following subsections, we describe the datasets used for the experimental evaluations of the proposed work along with the details of the preprocessing of the data.

A. Smart City Traffic Flow Prediction Case Study

Traffic flow prediction plays an important role in intelligent transportation management and route guidance. By collecting and analyzing the traffic information, many benefits can be brought from traffic predictions, for instance, it can help in relieving traffic congestion, reducing air pollution, providing road safety and secure traffic conditions [26]. However, since a vehicle's location is tightly bundled with its driver, it may threaten vehicles' location and trajectory privacy. An attacker can predict a driver's future location based on his vehicle's trajectory, or even infer the drivers' personal information, such as habits, health condition, income, and religious belief, according to their frequently visited places.

The aforementioned aspects of the task motivated us to deploy our proposed PSO-driven FL framework to the task

Algorithm 1 PSO for Parameter Optimization of Deep LSTM Models.

```
**Input**: A dataset. Values of acceleration constants ($c_1$ and $c_2$); $w; MaxIt$ (the maximum number of iterations to reach a good solution); and the range bounds $MinLayer$, $MaxLayer$, $MinNeurons$, $MaxNeurons$, $MaxLayerVelocity$ and $MaxNeuronVelocity$ (set as described in Table I).
**Output**: Near-optimal configuration in terms of $L$, $N$ and $E$ for the deep LSTM model.

**Begin**
1: Initialization
    a. Initialize $L_i$, $N_i$, and $E_i$ positions randomly for the number of layers, number of neurons, and number of epochs, respectively.
    b. Initialize $V_L$, $V_N$, and $V_E$ velocities randomly for the number of layers, number of neurons, and number of epochs, respectively.
    c. Define the fitness function (model accuracy for classification problem or root mean square error (RMSE) for the regression problem).
    d. Calculate the fitness value for each particle and set $pbest$ for each particle and $gbest$ for the population.
2: Repeat the following steps until the $gbest$ solution does not change anymore or the $MaxIt$ is reached.
    a. Update $L_i$, $N_i$, $E_i$, $V_L$, $V_N$, and $V_E$ in each particle according to the Equations (1) through (6).
    b. Calculate the fitness value for each particle. If the fitness value of the new location is better than $pbest$, the new location is updated to be the $pbest$ location.
    c. If the currently best particle in the population is better (e.g. in terms of Accuracy/RMSE) than the $gbest$, the best particle replaces the recorded $gbest$.
3: Return the near-optimal $L$, $N$ and $E$ for the LSTM model.
**End**
```

where the City Pulse EU FP7 project dataset [27] is used for traffic prediction. This dataset conveys the vehicular traffic volume collected from the city of Aarhus, Denmark, observed between two points for a set duration of time of 6 months. The dataset is divided into training and testing subsets. The training set is sampled into 18 different non-overlapped samples. Each sample is assigned to a client, which uses it to locally train a deep LSTM model. This means that the local training dataset of a client is never uploaded to the server. Instead, all the clients are coordinated by a central server, such that each client computes an update to the current global model maintained by the server, and only this update is communicated.

B. IIoT Predictive Maintenance Case Study

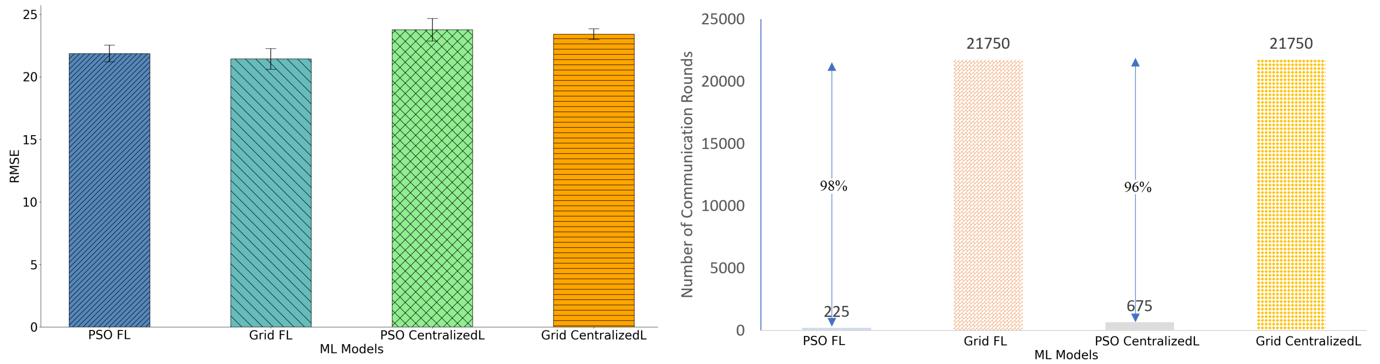
A predictive maintenance (PM) strategy uses ML methods to identify, monitor, and analyze system variables during an operation. Also, PM alerts operators to preemptively perform maintenance before a system failure occurs [28]. Being able to stay ahead of equipment shutdowns in a mine, steel mill, or factory, PM can reduce downtimes and maintenance costs for a busy enterprise. Such sensitive data leakage may cause a variety of cyber-attacks to Industrial IoT (IIoT) systems. One of the first successful attacks against industrial control systems was the Slammer worm, which infected two critical monitoring systems of a nuclear power plant in the U.S.A [29].

Our second case study is a proxy for IIoT services in which the real-time telemetry data, provided by Microsoft Azure Intelligence Gallery [30], created by data simulation methods collected from 100 machines in real time averaged over every hour collected during the year 2015 is used. The data consists of voltage, rotation, pressure, vibration measurements, error messages, historical maintenance records that include failures, and machine information such as type and age. The goal is to analyze predictive maintenance use-case problems caused by component failures such that the question "What is the probability that a machine will fail in the near future due to a failure of a certain component?" can be answered. Telemetry data comes with time-stamps that makes it suitable for calculating lagging features. In our experiments, 24 hours lag features are calculated. The prediction problem for our experiments is to estimate the probability that a machine will fail in the next 24 hours due to a certain component failure (component 1, 2, 3, or 4). All records within a 24 hours window before a failure of component 1 have $failure = comp1$, and so on for components 2, 3, and 4; all records not within 24 hours of a component failure have $failure = none$. The dataset is divided into training and testing subsets. The training dataset is sampled into 99 different non-overlapped samples (each sample represents the dataset for one machine) each sample is then assigned to a client that is used to train a deep LSTM model locally. All the clients are coordinated by a central server by updating the current global model maintained by the server.

V. Experimental Results

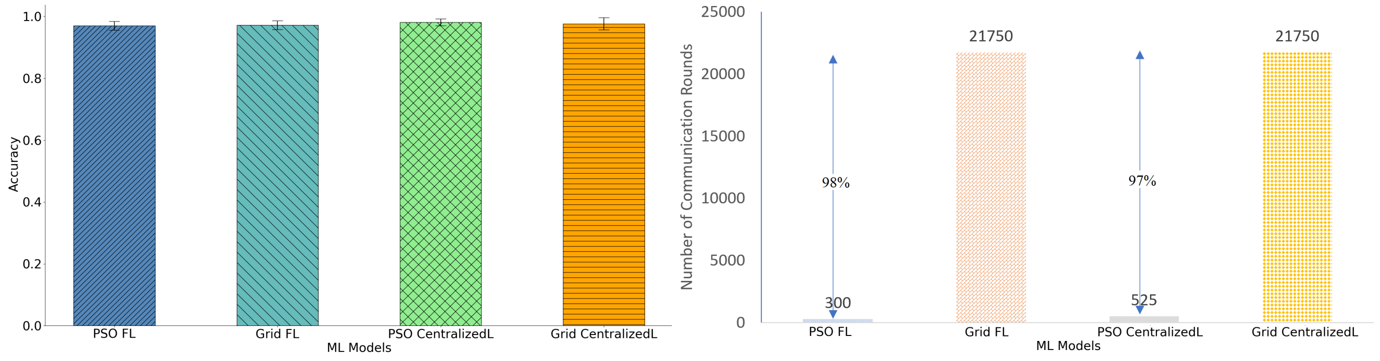
In our experiments, we selected two case studies to assess the performance of our proposed PSO-based parameter selection technique. We formulated the traffic prediction service as a regression problem, when LSTM is used to predict the number of cars, we use RMSE to measure LSTM model performance. While the predictive maintenance service as a classification problem to predict the probability that a machine will fail in the next 24 hours due to component failure, we use accuracy to measure LSTM model performance. We then set out to address the main goal of this paper which is to compare our proposed PSO-based parameter selection technique vis-à-vis the grid search technique in terms of finding the best L, N, and E for the LSTM models.

To evaluate and compare the grid search and PSO approaches, both the accuracy and the number of client-server communication rounds to explore the landscape of configurations to find the near-optimal parameters are considered. In the case of PSO, we consider the PSO technique with 5 particles. The algorithm terminates when the maximum number of iterations is reached or when there is no difference between the accuracies of two consecutive iterations. In our experiments, we also compare the proposed PSO against a grid search-based approach to find the best configurations for the models in the FL environment versus centralized learning.



(a) Error rate vs. Model (b) Number of Communication Rounds vs. Model

Fig. 1. Comparison between PSO and grid search in terms of **error rate** & **number of communication rounds** for ***smart city traffic prediction use-case***. The PSO-based technique reduces the number of clients-server communication rounds by 96–98% without adversely affecting the error rate of the models



(a) Accuracy vs. Model (b) Number of Communication Rounds vs. Model

Fig. 2. Comparison between PSO and grid search in terms of **accuracy** & **number of communication rounds** for ***IIoT predictive maintenance use-case***. The PSO-based technique reduces the number of clients-server communication rounds by 97–98% without adversely affecting the accuracy of the models

Centralized learning refers to the scenario in which users' data is uploaded to a centralized server for training, and then re-deploying an iterative model back to the user.

A. *Smart City Traffic Flow Prediction Results*

Figure 1-(a) shows the error rates of the best configurations using the proposed PSO and grid search in FL and centralized learning in traffic flow prediction use-case. This figure shows the RMSE as a 95% confidence interval for the best five configurations obtained by PSO and grid search approaches. Figure 1-(b) shows the number of clients-server communication rounds that need to convergence the globally best solution for FL models in in traffic flow prediction use-case.

B. *IIoT Predictive Maintenance Results*

Figure 2-(a) shows the accuracy of the best configurations using the proposed PSO and grid search in FL and centralized learning schemes in predictive maintenance use-case. This figure shows the accuracy as a 95% confidence interval for the best five configurations obtained by PSO and grid search approaches.

Figure 2-(b) shows the number of clients-server communication rounds that need to convergence the globally best solution for FL models in IIoT predictive maintenance use-case.

VI. Findings and Lessons Learned

Based on our results, our findings and lessons learned are summarized next.

1) Our PSO-based technique can return impressive performance for parameter tuning of deep LSTM models and improve upon the results of grid search.
2) Our proposed approach strives to decrease the exploration process of the landscape of configurations to find the near-optimal parameters without affecting the accuracy of the selected models, as Figures 1 and 2 indicate.
3) Figures 1 and 2 illustrate that the number of clients-server communication rounds can be massively decreased (to around 2%–4% of their original values returning a two-order of magnitude performance improvement) using our proposed PSO-based parameter value selection technique when compared with grid search method, while using PSO technique to find near-optimal configurations does not adversely affect the accuracy of the models trained in FL compared with centralizing learning approaches.
4) Figures 1-(a) and 2-(a) show that using the proposed PSO technique is not affecting the accuracy of the models trained in federated and centralized learning.
5) Figures 1-(b) and 2-(b) show that the number of clients-server communication rounds to explore the landscape of configurations to find the near-optimal parameters is

decreased by around two orders of magnitude (to around 2%–4% of the rounds) using the PSO-based approach compared to the grid search method in both decentralized FL-based approach and centralized learning approach.

VII. Conclusion and Future Work

Since the bottlenecks in the performance of the local models trained at the edge hinder the success of Federated Learning (FL), we have proposed in this paper a Particle Swarm Optimization (PSO)-based technique to optimize the hyper-parameter settings for the deep Long Short-Term Memory (LSTM) models trained at the edge in an FL environment. The parameters we optimized in the work include the number of hidden layers, the number of neurons in each layer for deep LSTM, and the number of epochs which is the number of training passes each client makes over its local dataset on each round. The proposed method is evaluated in two use-cases. In the first case study, we consider traffic prediction models as a proxy of smart city services. While in the second case study, we consider predictive maintenance models as a proxy for industrial IoT (IIoT) services. The results obtained show that the number of client-server communication rounds to explore the landscape of configurations to find the near-optimal parameter settings is greatly decreased by two orders of magnitude using the PSO-based approach compared to the grid search method. Our results also show that using the proposed PSO technique to find the near-optimal configurations without affecting the accuracy of the models in FL and centralized learning. As a future extension, we intend to explore the use of PSO to tune other deep learning parameters (e.g., batch size, learning rate) as well as the other parameters that are associated with FL, such as the fraction of number of clients that the server communicates within each communication round and the number of federated learning communication rounds.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, 2013.

[3] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning on medical data," *arXiv preprint arXiv:1811.12629*, 2018.

[4] B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou, "Parameters optimization of deep learning models using particle swarm optimization," in *2017 13th IWCMC*. IEEE, 2017, pp. 1285–1290.

[5] K. Swingler, *Applying neural networks: a practical guide*. Morgan Kaufmann, 1996.

[6] M. J. Berry and G. S. Linoff, *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons, 2004.

[7] K. Ahmad, M. L. Mekhalfi, N. Conci, F. Melgani, and F. D. Natale, "Ensemble of deep models for event recognition," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 2, pp. 1–20, 2018.

[8] K. Ahmad and N. Conci, "How deep features have improved event recognition in multimedia: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 2, pp. 1–27, 2019.

[9] 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*, 2017, pp. 1273–1282.

[10] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *arXiv preprint arXiv:1908.07873*, 2019.

[11] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "Federated optimization for heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 1, no. 2, p. 3, 2018.

[12] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[13] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[14] N. Tran, J.-G. Schneider, I. Weber, and A. Qin, "Hyper-parameter optimization in classification: To-do or not-to-do," *Pattern Recognition*, p. 107245, 2020.

[15] Y.-x. Wang, X.-z. Li, and Z.-y. Wang, "Parameters optimization of svm based on the swarm intelligence," in *Journal of Physics: Conference Series*, vol. 1437, no. 1. IOP Publishing, 2020, p. 012005.

[16] A. Tharwat and A. E. Hassanien, "Quantum-behaved particle swarm optimization for parameter optimization of support vector machine," *Journal of Classification*, vol. 36, no. 3, pp. 576–598, 2019.

[17] R. Rajalaxmi and E. Vidhya, "A mutated salp swarm algorithm for optimization of support vector machine parameters," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE, 2019, pp. 979–983.

[18] Y. M. M. Hassim and R. Ghazali, "Solving a classification task using functional link neural networks with modified artificial bee colony," in *2013 Ninth International Conference on Natural Computation (ICNC)*. IEEE, 2013, pp. 189–193.

[19] G. Zhou, H. Moayedi, M. Bahiraei, and Z. Lyu, "Employing artificial bee colony and particle swarm techniques for optimizing a neural network in prediction of heating and cooling loads of residential buildings," *Journal of Cleaner Production*, p. 120082, 2020.

[20] Z. G. Çam, S. Çimen, and T. Yıldırım, "Learning parameter optimization of multi-layer perceptron using artificial bee colony, genetic algorithm and particle swarm optimization," in *2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2015, pp. 329–332.

[21] S. Mirjalili, S. Z. M. Hashim, and H. M. Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11 125–11 137, 2012.

[22] F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm and Evolutionary Computation*, vol. 49, pp. 62–74, 2019.

[23] T. Serizawa and H. Fujita, "Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization," *arXiv preprint arXiv:2001.05670*, 2020.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, ISSN: null.

[25] M. Clerc, *Particle Swarm Optimization*. John Wiley & Sons, google-Books-ID: Slee72idZ8EC.

[26] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," vol. 16, no. 2, pp. 865–873.

[27] CityPulse smart city datasets. http://iot.ee.surrey.ac.uk:8080/.

[28] B. Qolomany, I. Mohammed, A. Al-Fuqaha, M. Guizani, and J. Qadir, "Trust-based cloud machine learning model selection for industrial IoT and smart city services," *arXiv:2008.05042 [cs]*. [Online]. Available: http://arxiv.org/abs/2008.05042

[29] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, ISSN: 0738-100X.

[30] Azure AI gallery. [Online]. Available: https://gallery.azure.ai/