Particle Swarm Optimized Federated Learning For Securing IoT Devices

Pushkar Kishore
Dept. of C.S.E.
NIT Rourkela
Odisha, India
518CS1002@nitrkl.ac.in

Swadhin Kumar Barisal
Dept. of C.S.E.
NIT Rourkela
Odisha, India
swadhinbarisal@gmail.com

Durga Prasad Mohapatra
Dept. of C.S.E.
NIT Rourkela
Odisha, India
durga@nitrkl.ac.in

2021 IEEE Globecom Workshops (GC Wkshps) | 978-1-6654-2390-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/GCWkshps52748.2021.9681946

Abstract—**Federated learning (FL) focuses on interpreting optimization, privacy, and communication but pays little consideration to enhance training and results on the edge devices. The major challenge on these Internet of Things (IoT) devices is efficient training and inference. Another considerable challenge is securing IoT devices for a long time. This paper resolves it by selecting appropriate parameters for building a local machine learning or deep learning (ML/DL) model. Appropriate parameters will make the model's training less computationally expensive and secure the edge or IoT device. So, we propose a particle swarm optimization (PSO) method to optimize the hyperparameter environments for the bounded DL model in an FL environment. First, we select the 2-gram represented Application Programming Interface (API) calls of the malicious and benign instances for the dataset's feature. Then, API calls of the sample are represented using 2-gram, and their frequency fills the dataset's rows. Later, we represent the sample's feature in a grayscale image and apply the LeNet-5 model. Our experiment indicates that PSO efficiently tunes the hyperparameters of LeNet-5 compared to the grid search method. The near-optimal parameters for FL do not affect the model's accuracy.**

Index Terms—**Edge devices, API, Grayscale, LeNet-5**

I. Introduction

Internet of Things (IoT) is a network of devices that continuously send and receive data over the internet. Newman [1] reported that the number of IoT devices is expected to reach 41 billion by 2027. Securing IoT devices is difficult even for manufacturers and companies. The fast-evolving nature of IoT device types, traffic patterns, and cyber-attacks continuously make it challenging to ensure security on edge devices.

There have been tremendous improvements in the processing power of edge devices, and concern about data privacy and security has paved the path for federated learning (FL) [2]. FL helps in building a machine learning model working on multiple IoT devices. The significant advantage of the FL lies in enabling global model training without infringing data privacy or weakening security. The FL's performance depends on the edge device's capability to train the models locally and infer test data with considerable accuracy. However, the efficacy of the models is highly dependent on the selected hyperparameters. For example, the deep neural network (DNN) performance dramatically varies with the values of parameters like the number of neurons, epochs, and hidden layers [3]. So, the principal challenge for the training model at the edge is deciding on proper hyperparameter configuration.

FL has some challenges like ensuring data privacy, providing efficient communication, and building a global model from distributed and heterogeneous data. Generally, devices are running ML models trained on the local dataset without exploring optimal values for hyperparameters. Some heuristics are present [3], but no proper theory defines the structure of models for specific applications.

This paper proposes a particle swarm optimization (PSO) [4] based technique to optimize the hyperparameter settings for the local DL model in an FL environment. In detail, we utilize PSO to optimize 6 parameters namely, (i) optimizer, (ii) activation, (iii) batch size, (iv) neurons, (v) epochs and (vi) patience. To ensure the device's security, we select 2-gram Application Programming Interface (API) calls and create the dataset. Image processing based technique named LeNet-5 [5] is used for training which has its hyperparameter configured by PSO. PSO and other heuristic techniques work on randomized guesses to find a solution. We use PSO as an optimization method for our research due to its efficiency, simplicity, and fast convergence. Due to the above features, PSO is used in diverse application domains. In our work, we optimize the FL model by searching possible space of hyperparameters instead of assuming fixed parameters for IoT devices.

The main contributions of this work are:

1) We propose a secured framework for edge or IoT devices in which the model is trained using the hyperparameters selected using PSO.
2) We define the dataset feature using n-gram API calls and find out the best value of n.
3) We compare the proposed PSO based optimization approach with the grid search technique and figure out the best models running on edge devices.

The remaining part of this paper is organized as follows: Section II discusses related state-of-the-art work, Section III presents the proposed architecture, Section IV describes the experimental setup and its results, Section V discusses the comparison with related state-of-the-art works, and Section VI concludes the article with future directions.

II. Related Work

Many optimization techniques had been proposed in the past to train the ML models in the FL environment with unbalanced and highly distributed data. Mcmahan et al. [6] developed a synchronous scheme termed as FedAvg and used it for optimization of the global model. FedAvg technique was uncertain about the convergence in the presence of the heterogeneous data [7]. Therefore, Sahu et al. [7] modified the version of FedAvg and named it as FedProx. FedProx contained proximal data in the objective function, which helped keep the model stable in heterogeneous data. Smith et al. [8] proposed a multi-task learning based solution named MOCHA which could learn related models for each node parallelly.

Tran et al. [9] studied the impact of parameters optimization on the performance of traditional classification algorithms. Zhou et al. [10] applied bee colony and PSO optimization techniques separately and in combination to optimize the weights and biases of multi-layer perceptron NN. However, parameters like the number of hidden layers or learning rate are equally crucial for defining the performance and feasibility of NN. Junior et al. [11] proposed psoCNN, which could find an optimal convolutional neural network (CNN) architecture for image classification. Finally, Serizawa et al. [12] proposed a linearly decreasing weight particle swarm optimization for optimizing the parameters of the CNN architecture.

Peiravian et al. [13] applied a machine learning based approach on static features. They extracted two types of features, namely user permissions and API function call. Yuan et al. [14] proposed 'Droid-Detector', a hybrid approach for android malware detection. The results showed that deep learning was suitable for characterizing Android malware and especially effective with the availability of more training data. Hou et al. [15] proposed a dynamic analysis technique, namely component traversal, that can execute the android application's code routines. Then, they constructed weighted directed graphs and applied the deep learning technique.

III. Proposed Architecture

This section illustrates the overall mechanism of our model. Fig. 1 represents the architecture of our proposed model [1]. The proposed model is divided into four components: (i) Extracting Dynamic API calls (ii) Create n-gram vectors of API calls (iii) Select the best configuration of hyperparameter using PSO and grid technique (iv) Apply LeNet-5 to classify the instances. The description of the above steps are discussed below:

A. Extracting Dynamic API calls

Dynamic analysis is intended to accumulate features on the application during run time. We selected API calls to provide tools and procedures for building edge applications. Static analysis is adequate for classifying malicious applications but alone cannot predict complex malware [16]. Thus, malicious samples executing their payload after the start of the application will remain stealth from static analysis.

[1]https://github.com/pushkarkishore/PSO-FL-Edge

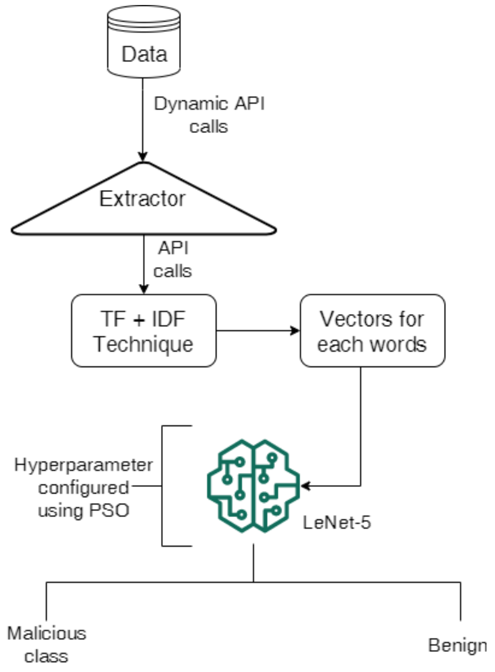

Fig. 1. Proposed Architecture

Owing to the above points, we extracted dynamic API calls for our work.

B. Create n-gram vectors of API calls

The API calls collected from the application are arranged into multiple lists using the n-gram technique. An n-gram is a contiguous sequence of n-items extracted from a list of words or texts in probabilistic techniques. We are uncertain about the value of n, which will effectively differentiate between malicious and benign apps. Thus, the value of n was varied from 1 to 3 for finding the appropriate one. Upon experimenting with n's values, $n = 2$ is considered for further analysis. The experimental results are provided in Section IV. The frequency of the individual 2-grams vector is evaluated within the list of the 2-gram API calls. Instead of using a count vectorizer, term frequency-inverse document frequency (TF-IDF) was used.
We represented the instance as an image with the 2-grams defining feature, and the frequency of 2-grams determines the pixel's intensity. We kept all the 2-gram API calls that represented the malicious or benign samples. Gray-scale is used since its representation needs the intensity of the pixels, and the same intensity pixels are correlated. Our 2-gram API calls have their frequency indicating the intensity of the pixel. The reliance on the pixel's intensity ensures that the image will not break at its edge, and no dummy relation exists between two diagonal pixels having different intensities. If we have used the sequential representation, then the image will have no issue like breaking and dummy relations. However, the sample's behavior is well recognized using short consecutive calls instead of long ones. Thus, we preferred short consecutive API calls of size two in our experiment.

C. Select the best configuration of hyperparameterS using PSO and grid technique

In PSO, swarm depicts the potential solution of the problem and regularly communicates with each other. Each swarm is associated with its position, velocity, and fitness value. We initialized few parameters like an optimizer, activation, batch size, neurons, epochs, and patience. Then, we searched for optima in later generations. The considered swarm's and all other swarm's best positions were determined at each iteration. The best value is the global best, and each swarm updates its velocity and position based on the swarm's best position. The velocity ($V_{L,i}^{t+1}$) and position (L_i^{t+1}) of each swarm for all parameters like optimizer, activation and so on are updated according to Equations 1 and 2.

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

where, V_L is the velocity, L_i^{best} is the swarm's best local value and G^{Lbest} is the best global value, c_1 and c_2 are acceleration coefficients, w is the inertia coefficient and $rand$ is a uniform random value between 0 and 1.

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

Table 1 shows the hyperparameters to be optimized by PSO and grid method.

D. Apply LeNet-5 to classify the benign or malicious instances

The LeNet-5 model is selected due to its lightweight architecture compared to AlexNet, GoogLeNet and so on.

IV. Experimental setup and results

In this section, we describe datasets, setups, and results of our experiment.

TABLE I
HYPERPARAMETER OPTIONS FOR OPTIMIZING PSO AND GRID

Parameters	**Value**
Optimizer used	adam or sgd
Activation function	relu or tanh
Batch size	16 or 32
Number of neurons	[80,100] step size varied
Number of epochs	[20, 25] increment by 1
Epochs to wait before early stop (Patience)	[2,5]

TABLE II
MALWARE DATASET

Malware family	**Number of samples**
Virus	2556
Trojan	168
Spyware	31
Risktool	579
Scareware	37
Ransomware	46
Downloader	44
SMSmalware	42
Adware	390
Dropper	75
Rootkit	15
Total number of malware	3983

A. Setup

We implemented our experiments on a single machine. The CPU version is Intel i5-3470 @ 3.20GHz and the RAM is 16 GB. To evaluate our proposed model's performance, we replaced few dataset [2] samples with samples targetting edge devices and selected 8287 instances. Out of those instances, 4304 are benign, and 3983 are malicious samples. In contrast, all the instances are considered to be at a centralized server to find the hyperparameter configuration, suitable for better classification accuracy. The details of the malware families are discussed in Table II.

B. n-gram vectors of API calls

For finalizing n, we tested the centralized model's detection true positive rate (TPR) on all the samples and selected one with the highest TPR. Detector's TPR was evaluated by fixing FPR at 10^{-3} and executing 5-fold cross-validation. Thus, the value of n was varied from 1 to 3. For n=1, TPR was 98 %, indicating that n=1 is insufficient for detection. For $n \varepsilon \{2, 3\}$, the TPR is highest (99%). The TPR does not improve while incrementing n to 3 from 2 since overfitting nullifies the benefit of existing features with additional features. Furthermore, since the number of features scales exponentially with n, we select the value of $n = 2$.

[2]https://www.kaggle.com/goorax/static-analysis-of-android-malware-of-2017

C. Best configuration of hyperparameter using PSO and grid technique

Our security provider service is a detection and classification problem. LeNet-5 is used to detect malicious instances and predict its family. We then address this paper's primary goal: to compare our proposed PSO-based parameter selection technique with the grid search technique. We considered the accuracy and number of client-server communication rounds to determine the near-optimal parameters using grid search, and PSO approaches. In the case of PSO, we considered the PSO with five particles. The value of coefficients and inertia was two and 0.2, respectively. Finally, we compared the proposed PSO with grid search for finding the best hyperparameter configuration for models in FL environment vs. centralized learning. In centralized learning, the user's data is uploaded to a centralized server for training and redeploying an iterative model on the user end.

D. LeNet-5 model's detection and classification result

Fig. 2 shows the detection accuracy of the best configurations using the proposed PSO and grid search in FL and centralized learning. Detection accuracy is evaluated by dividing the count of correct malicious and benign predictions by the set of instances. We achieve considerable accuracy approaching the grid using PSO in the case of FL and centralized. The difference between detection accuracy of PSO FL and Grid FL is 0.4% while 0.3% between PSO centralized and Grid centralized. Thus, the proposed PSO technique does not affect the accuracy of the models in federated and centralized. Fig. 3 shows the number of client-server rounds required for converging to a globally best solution in security provider use case. The rounds differ by 89.6% in the case of FL while

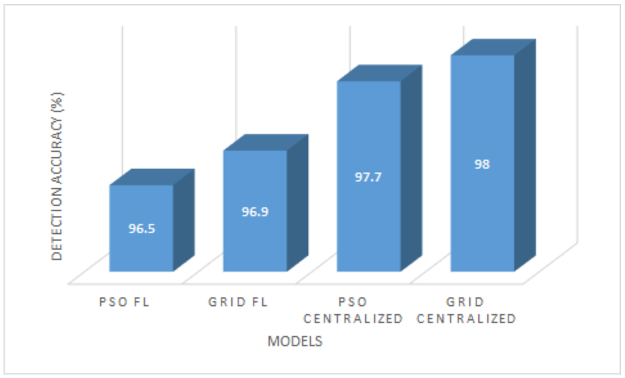

Fig. 2. Detection accuracy vs. Model

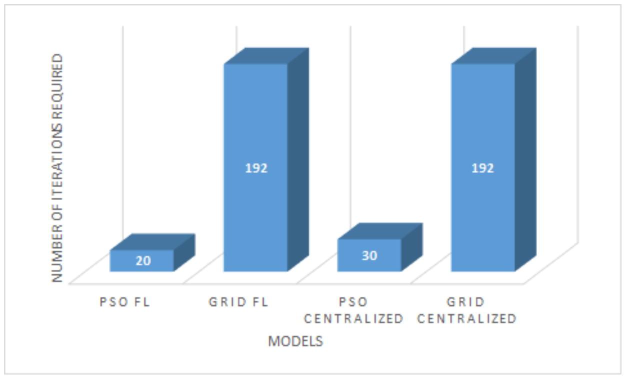

Fig. 3. Number of iterations vs. Model for detection

by 84.4% in centralized learning. Fig. 4 shows classification accuracy of the best configurations using the proposed PSO and grid search in FL and centralized. Classification accuracy is determined by the percentage of samples classified accurately out of all samples. In the case of classification, we consider all families of samples, i.e., virus, trojan, rootkit, benign, etc. The difference between the accuracy of PSO FL and Grid FL is 2% and similar for PSO centralized and grid centralized.
Thus, our PSO-based technique can provide adequate performance for parameter tuning of the LeNet-5 model. On the other hand, Fig. 2, 3 and 4 indicate that our proposed approach diminishes the exploration process of finding proper hyperparameters by suggesting near-optimal parameters with the least affected accuracy. The number of client-server communication rounds is decreased to around 10-15% of their values in grid FL and centralized model. Therefore, the accuracy is not adversely affected when the client-server communication rounds are reduced drastically. Table III mentions the final hyperparameters used for generating result.
The batch size required for federated is higher than centralized as edge devices need a larger batch size for better accuracy due to the smaller dataset available with them. Also, higher epochs help FL devices to get trained with generalized accuracy. SGD is suitable for FL devices since it computes on a small subset of data and is efficient for low memory or processing power devices. On the other side, ADAM is fast than SGD and helpful to centralized servers with more significant memory and processing power.

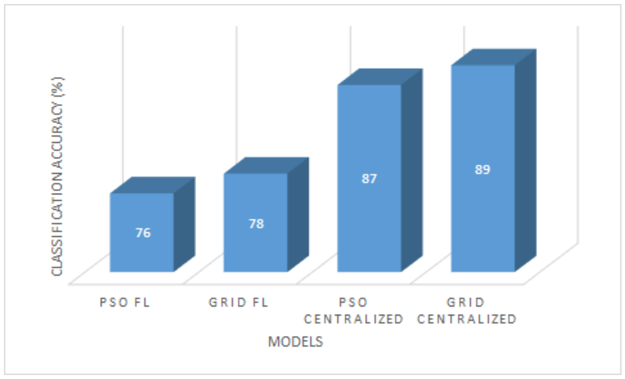

Fig. 4. Classification accuracy vs. Model

V. Comparison with Related Works

Our objective is to train the model on the edge devices and secure them. Table IV compares the detection performance of our proposed PSO FL model with related works. In Table IV, prec. is the precision of the model while rec. is the recall of the model. We implemented the techniques like support vector machine, J48, bagging along with few deep learning techniques on the similar dataset considered for our proposed approach. Our

TABLE III
Parameters used for FL and centralized

Parameter	FL	Centralized
Optimizer used	sgd	adam
Activation function	relu	tanh
Batch size	32	16
Number of neurons	120	80
Number of epochs	25	20
Epochs to wait before early stop (Patience)	2	5

TABLE IV
DETECTION PERFORMANCE OF OUR VS. OTHERS

Method	Acc.(%)	Prec.(%)	Rec.(%)
SVM [13]	85.75	81.7	85.7
J48 [13]	84.46	80.6	82.8
Bagging [13]	86.39	84.9	84.1
DD-Static [14]	89.03	90.39	89.04
DD-Dynamic [14]	71.25	72.59	71.25
DD-Static+Dynamic [14]	86.62	85.6	87.6
DD-DL [15]	83.68	83.69	83.36
PSO FL (Proposed)	**96.5**	**94.2**	**95.0**

proposed model has the highest accuracy, precision, and recall compared to other models. In the case of techniques like SVM, J48, and bagging [13], inappropriate kernel and poor choice of hyperparameters decreased the model's accuracy. In the case of DD-Static, DD-Dynamic, or DD-Static+Dynamic [14], training is more difficult due to the calculation of the energy gradient function on edge devices. Weight adjustment is lengthy due to the giant stack of restricted boltzmann machines. DD-DL [15] is ineffective due to consideration of system calls which the obfuscated malicious instances like ransomware can hide. Our proposed model is effective due to the proper selection of hyperparameter configuration with a lightweight training model that needs limited data and shorter training time.

VI. CONCLUSION AND FUTURE WORK

The bottleneck in the performance of the local models trained at edge benefit of federated learning. We proposed particle swarm optimization to optimize the hyperparameter settings for the LeNet-5 model trained at the edge in the FL environment. LeNet-5 is applied to the dataset defined to secure edge devices. The dataset's features were defined using 2-gram API calls, and instances' values were normalized using the term frequency-inverse document frequency. The results showed that the number of client-server communication rounds to explore near-optimal hyper-parameter configuration is decreased by around 85% using PSO compared to grid. Even the accuracy was not adversely affected for FL and centralized learning while reducing the client-server communications. In the future, a few more deep learning parameters will be optimized for quick and efficient training at edge devices.

REFERENCES

[1] P. Newman, "The internet of things 2020: Here's what over 400 iot decision-makers say about the future of enterprise connectivity and how iot companies can use it to grow revenue," *Business Insider*, pp. 1–6, 2020.

[2] 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.

[3] 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 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 1285–1290.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[5] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[6] 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.

[7] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 3, p. 3, 2018.

[8] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," *arXiv preprint arXiv:1705.10467*, 2017.

[9] N. Tran, J.-G. Schneider, I. Weber, and A. K. Qin, "Hyper-parameter optimization in classification: To-do or not-to-do," *Pattern Recognition*, vol. 103, p. 107245, 2020.

[10] 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*, vol. 254, p. 120082, 2020.

[11] 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.

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

[13] N. Peiravian and X. Zhu, "Machine learning for android malware detection using permission and api calls," in *2013 IEEE 25th international conference on tools with artificial intelligence*. IEEE, 2013, pp. 300–305.

[14] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[15] S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*. IEEE, 2016, pp. 104–111.

[16] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, "The evolution of android malware and android analysis techniques," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, pp. 1–41, 2017.