2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom) | 978-1-6654-9825-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/MEDITCOM55741.2022.9928614

Sine Cosine Algorithm for Reducing Communication Costs of Federated Learning

Ammar Kamal Abasi, Moayad Aloqaily, Mohsen Guizani, and Fakhri Karray

Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates
E-mails: {ammar.abasi; moayad.aloqaily }@mbzuai.ac.ae, mguizani@ieee.org, fakhri.karray@mbzuai.ac.ae

***Abstract*—Federated Learning (FL) is a Machine Learning (ML) setting in which several clients (e.g., mobile devices) train a model cooperatively under the direction of a central server (e.g., cloud server), while training data is decentralized. Due to the fact that FL clients frequently have restricted transmission capacity, communication among clients and servers needs to be reduced to enhance presentation. FL clients frequently employ Wi-Fi and must interact in Unstable Network Environments (UNE). Existing FL aggregation techniques send and receive a huge number of weights, which dramatically reduces the accuracy of the UNE. In this paper, we propose a Federated Sine Cosine Algorithm (FedSCA) to reduce data communication by transferring score principles rather than all client models' weights and utilizing the Sine Cosine Algorithm (SCA) mechanism as a weight updating technique to improve the clients' models. This paper reveals that using FedSCA significantly decreases the quantity of data utilized in network communication and increases the global model's accuracy by an average of 9.87% over FedAvg and 2.29% over Federated Particle Swarm Optimization (FedPSO). Moreover, in studies conducted on an unstable network, it demonstrated a 4.3% improvement in comparison to accuracy loss in existing algorithms.**

***Index Terms*—Sine Cosine Algorithm (SCA), Convolutional Neural Network (CNN), Federated Learning, Deep Learning, Optimization.**

I. Introduction

Machine Learning (ML) models are getting more and more complex with the increase in the size of datasets. This is why training ML models has become a time-consuming and challenging task for data scientists [1]. The currently available ML techniques are designed to be used in highly controlled environments, such as data centers, in which the distribution of data between machines occurs in a balanced and Independent and Identically Distributed (IID) fashion and where high-throughput networks are available. Recently, Federated Learning (FL) [2] has been suggested as an alternate ML setting: a centralized server generates the training of a shared global model from a federation of participating devices (or clients). The number of participating devices is often relatively high, and their internet connections are either poor or unreliable.

A primary motivating example for FL emerges when the training data is derived from user interactions in mobile applications [3]. Sharing prediction models is made possible by FL's ability to keep all training data on a mobile device, hence eliminating the need to store training data in a cloud-based service. To update a global model, users' mobile devices are used as nodes for computation according to their data, which is stored locally on their devices. By moving model training to the mobile device, this expands the applicability of local predictive models.

In fact, there are four considerations when leveraging mobile device data for FL. First, collecting or keeping private information raises the probability of data breaches. Second, mobile device processors lack the computational capacity required for ML (i.e., low computational capability). Third, as mobile devices cannot connect to wired networks, maintaining a stable network environment is difficult because Wi-Fi connections are prevalent. Finally, there is a considerable cost associated with acquiring and placing a considerable quantity of source data on a server to be stored[4].

In order to design a successful ML model, especially for Artificial Neural Network (ANN) weights using mobile data, it is essential to minimize the quantity of the gathered data, enhance the security of the data collected, improve the stability of an Unstable Network Environment (UNE), and reduce the number of training parameters. As the ANN is trained, its back-propagation mechanism adjusts its weights. In order to address the aforementioned challenges, the research on FL has consistently progressed, enabling the use of large amounts of data on mobile devices [5]. The FL technique safeguards private information by keeping the data in the local IoT device and not share it with a centralized repository [6]. Furthermore, FL keeps transmission costs to a minimum by merely uploading the learned models to the server rather than uploading enormous volumes of source data.

In standard ANN models, the calculation time is significantly greater than the communication time, hence several strategies are employed to minimize the calculation time, including the use of Graphics Processing Unit (GPU) accelerators and the connection of multiple GPUs. Communication in FL, on the other hand, takes more time than computing. Therefore, FL's performance could be increased by reducing the time for a network connection. FL requires Wi-Fi connections and connected chargers due to UNE issues [7]. Therefore, in order to lower FL's communication costs, it is required to increase network transmission speed and address UNE issues. FedAvg is the most common method of incorporating FL into a model. The goal of this paper is to adapt the Sine Cosine Algorithm (SCA) [8] strategy to speed up model updates. Due to the stochastic mechanism used in SCA, the SCA requires a large number of iterations, which is in line with ML's strategy of learning via many reruns [9]. The SCA is well-suited to settings that are both heterogeneous and dynamic environments.

This paper aims to reduce network communication costs. As a way to update global models by using SCA in conjunction with FL communications. We have created a Federated SCA (FedSCA) model, which uses scores like accuracy and loss as a objective function. The FedSCA network cost and accuracy are also evaluated. By comparison with other algorithms, FedSCA was shown to be 9.87% and 2.29% more accurate on average than FedAvg and FedPSO, respectively. Moreover, we used an UNE to evaluate FedSCA, and the proposed FedSCA showed a 4.3% decrease in accuracy loss when compared to existing algorithms.

The remainder of the paper is organized as follows: Section II examines past research utilizing SCA and FL. Section III discusses how the proposed method transmits the learned model from the client to the server. The assessment of our proposed method is offered in Section IV, followed by the paper's conclusion in Section V.

II. Background and Related Work

A. Federated Learning (FL)

Formally, the objective of a Federated Learning (FL) algorithm is to solve a distributed (federated) supervised learning optimization problem [10]. A collection of $k = 1, 2, ..., K$ clients seek to concurrently optimize the global model parameters $w \in \mathbb{R}^d$ by utilizing their local data $\{X_k, Y_k\}$ of size N_k from N samples. Clients' local losses, denoted by $l_k(w, X_k, Y_k)$, are pooled to minimize a global cost function with a finite sum:

$$min_{w \in \mathbb{R}^d} \sum_{k=1}^{K} \frac{N_k}{N} l_k(w, X_k, Y_k), \tag{1}$$

FL is a learning approach for distributed datasets suggested by Konecn [2]. It prevents data leakage when training a model from datasets dispersed across several devices [11]. FL is useful since it enhances privacy and decreases communication expenses [12]. Artificial Neural Network (ANN) models can learn through federated learning without data or personal information intrusions. Network traffic and storage costs rise when data is sent between several devices and then stored on a centralized server. FL greatly minimizes communication costs by transmitting just model-training-derived weights. The FL procedure is outlined in Figure 1 and described below.

1) The learning global model is sent from the server to all selected clients.
2) The models received on devices are trained using client data.
3) Each client delivers the server its trained local model.
4) The server processes and integrates the obtained models into a single updated model.
5) This step is performed for each of the client's updated models from the server.

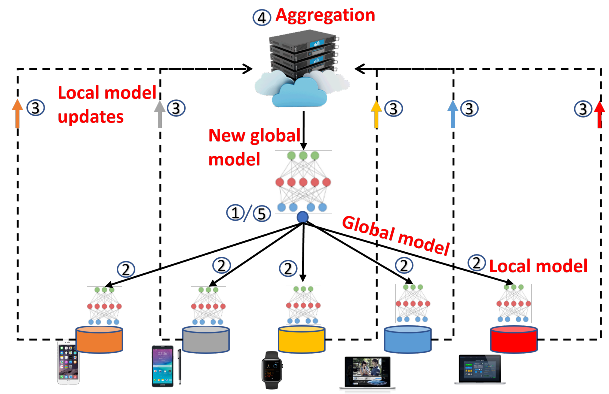

Fig. 1. The Federated Learning protocol.

Federated Stochastic Gradient Descent (FedSGD) is used in a large number of FL studies. As shown in Figure 1, the fourth step is accomplished by using FedSGD and Federated Averaging (FedAvg). FedAvg is an algorithm that was developed by McMahan [2] to address some of the limitations of FedSGD, such as being unable to scale well with large datasets and update server-collected model data. All the clients' parameters are averaged together to produce global model parameters. When FedSGD collects weights from the server, it returns them to the client. In this method, the global model weights are changed to construct a global model once the gradient is submitted to the server to determine the average gradient across clients. FedAvg leverages FedSGD and mini-batches to immediately update models on the client side while the server averages weights to produce a new global model.

FL implies a distributed mobile device environment. Mobile devices have the drawback of learning in a wireless network environment rather than a stable wired network connection [7]. Because of unreliable network conditions, a trained model may not be able to transfer its entire dataset when it is submitted to the network for evaluation.

B. Sine Cosine Algorithm (SCA)

Generally, random solutions are usually used as a starting point for population-based optimization algorithms [13]. The objective function and search guidelines form the basis of an optimization process, which continually evaluates and improves this random set of solutions [14]. There is no guarantee that a solution will be obtained in a single run due to the fact that population-based optimization algorithms search for optimum solutions to optimization problems in a random way (i.e., stochastic) [15]. However, the possibility of finding the global optimum increases as the optimization steps (iterations) and number of random solutions increase. Although stochastic population-based optimization has many algorithmic variants, the optimization process is commonly separated into two stages: exploration and exploitation [16]. Optimization algorithms first combine random solutions with a high degree of unpredictability in order to obtain interesting regions in the search space. Exploration and exploitation are two distinct phases, and random solutions are progressively altered throughout exploitation.

SCA is a new meta-heuristic algorithm that employs the sine and cosine functions' mathematical features. In 2015, Mirjalili developed this algorithm [8]. This population-based optimization algorithm starts with a random distribution of solutions. The following equations are then applied to each individual solution.

$$C_{X_i^{itr+1}} = C_{X_i^{itr}} + C_1 \times \sin(C_2) \times \mid C_3 G_{B_i^{itr}} - C_{X_i^{itr}} \mid \quad (2)$$

$$C_{X_i^{itr+1}} = C_{X_i^{itr}} + C_1 \times \cos(C_2) \times \mid C_3 G_{B_i^{itr}} - C_{X_i^{itr}} \mid \quad (3)$$

where $C_{X_i^{itr}}$ is a current solution at itr iteration in the i_{th} dimension, $G_{B_i^{itr}}$ is the best solution attained up to iteration itr. The values of C_2 and C_3 are completely random. The search space region around the current solution can be determined by the coefficient C_1. There is no definitive boundary between $G_{B_i^{itr}}$ and $C_{X_i^{itr}}$ in the search space. Using the C_1 parameter, it is easier to explore and exploit the search space during the search. The coefficient C_1 is devoted to exploring the search space in half of the maximum number of possible iterations and then uses the second half of iteration count for exploiting the search space. C_1 is defined in terms of the following mathematical formula:

$$c_1 = a - itr_i \frac{a}{Max_{itr}} \quad (4)$$

The maximum number of SCA iterations Max_{itr} is used as a termination criteria, and a is a constant (two in this paper). It is possible for C_1 to point towards or outside $G_{B_i^{itr}}$, depending on the present solution's moment direction. There are two weights in C_1 one for exploring ($C_1 >1$) and one for exploiting ($C_1 < 1$). C_1 also avoids premature convergence at the end of each iteration. A smooth transition from sine to cosine functions, and vice versa, can be achieved using the coefficient C_1. Here is how SCA makes use of the previous two equations:

$$C_{X_i^{itr+1}} = \begin{cases} C_{X_i^{itr}} + C_1 \times \sin(C_2) \times \mid C_3 G_{B_i^{itr}} - C_{X_i^{itr}} \mid & r_4 < 0.5 \\ C_{X_i^{itr}} + C_1 \times \cos(C_2) \times \mid C_3 G_{B_i^{itr}} - C_{X_i^{itr}} \mid & r_4 \geq 0.5 \end{cases} \quad (5)$$

The SCA algorithm's phases are explained in depth in Algorithm 1.

C. *Related Work*

There is a lot of work on how to improve FL performance through better communication with clients. Numerous problems arise when the UNE of mobile devices is used, including group shifting, significant server load, frequent node crashes, and increasing latency. Furthermore, multi-layer models have been employed to enhance learning accuracy. However, as the layers become deeper, the number of weights for the nodes also increases. FL is restricted by the amount of data because it increases the size of the network transmission between the server and the client.

Algorithm 1: Pseudocode of the SCA algorithm

```
Input: the SCA parameters (a, c1, Number of dimensions, Number of solutions,
   Number of iterations Max_itr).
Create the population of solutions.
while The current iteration itr_i less than the maximum number of iterations
   Max_itr do
    Compute the objective function for each solution.
    Determine the best solution G_{B_i^itr}.
    Compute c1 using Eq (4).
    Generate c2, c3, and c4.
    for each solution C_{X_itr} do
        for each decision variable i in C_{X_itr} do
            Update the current solution using Eq (5).
        end for
    end for
end while
Output: Return G_{B_i^itr}, which is the best solution.
```

There are two ways to reduce the uplink communication costs proposed by Konecny et al. [5] including: sketched updates, by learning the full model update and then compressing it using a combination of quantization, random rotation, and subsampling prior to sending it to the server; and structured updates, by learning an update from a smaller set of variables in a more restricted space, e.g., by using a low-rank space or random masks. The proposed methods by [5] achieved lower communication costs by two orders of magnitude on recurrent and convolutional networks.

Prior to the work presented in [5], the majority of past research has concentrated on improving client communication and global optimization in FL. Data transmission in FL's UNE has never been studied in depth. In addition, the SCA has never been used to increase the global model performance through improved network communication.

III. Federated Sine Cosine Algorithm

Adding extra model layers is a common way to improve the accuracy of Artificial Neural Network (ANN) models. An ANN with many parameterized layers is known as a "deep Neural Network". The number of weight parameters that must be learned grows as the number of layers increases. A considerable rise in network communication costs occurs when a model trained on a client is moved to a server in a global FL. Consequently, we propose the FedSCA approach to transport a model, independent of its size, while delivering the best loss or accuracy (i.e., score) to the server.

Before discussing the FedSCA, we will explain the FL algorithm used in previous research (i.e., FedAvg). Algorithm 2 shows the implementation of the FedAvg in FL. Line 4 selects the client who will participate in the round. Finally, the client's weight values are obtained in Lines 5 and 6. Weights acquired in Line 8 are averaged, and the global model weights are computed when the step in Line 8 has been completed. The client receives the server's global weights in Lines 10–13.

In the proposed algorithm, the FedSCA gets the model weights from the client who provided the highest score, avoiding the need to receive model weights from all clients. Figure 3 depicts the weight updating process of FedSCA. Using client training's lowest loss value, the best score is

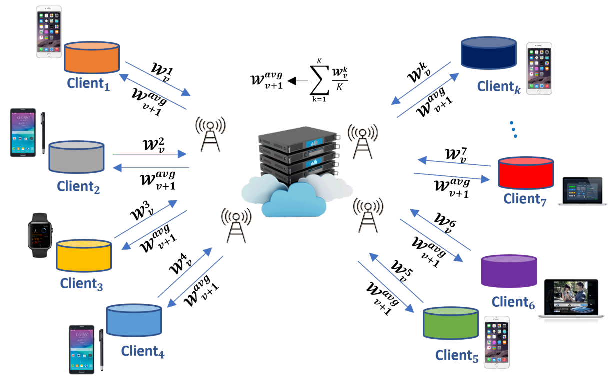

Fig. 2. Weighted aggregation processes like FedAvg receives an average of the W_t values that K clients transmit to the server and send an average of the updated $W_t + 1$ weights back to clients.

Algorithm 2: Pseudocode of the FedAvg algorithm

```
1: Function CLOUDSERVEREXECUTES :
2: Initialize global model (w_0).
3: for each iteration itr = 1, 2, 3...Max_itr do
4:     S_itr= set of maximum clients.
5:     for each client k ∈ S do
6:         w^t + 1=LocalClientUpdate(k,w^t)
7:     end for
8:     w^t + 1=average the weights
9: end for
10: Function LocalClientUpdate(k,w^t) :
11: for each client epoch i=1,2,3...E do
12:     Execute the learning procedure to client k.
13: end for
14: update w.
15: return w to server.
```

achieved. Four bytes are all that is needed for this loss value. As a result, FedSCA adjusts its weighted array using the current solution value for each member of the optimum model.

In light of the fact that the ANN weight values have been updated in Eq (5), the FedSCA weight values can be represented as follows.

$$C_{w_i^{itr+1}} = \begin{cases} C_{w_i^{itr}} + C_1 \times \sin(C_2)\times \mid C_3 G_{Bw_i^{itr}} - C_{w_i^{itr}} \mid & r_4 < 0.5 \\ C_{w_i^{itr}} + C_1 \times \cos(C_2)\times \mid C_3 G_{Bw_i^{itr}} - C_{w_i^{itr}} \mid & r_4 \geq 0.5 \end{cases} \quad (6)$$

, where $C_{w_i^{itr+1}}$ is denoted to next layer (l) weight value, $C_{w_i^{itr}}$ is the current layer (l) weight value, and $G_{Bw_i^{itr}}$ is the global model layers' weight values.

In Eq (6), the current ANN solution contains a weighted value assigned to each layer. Adding $C_{X_i^{itr}}$ to the prior step weight w^{t-1} yields the current step weight w^t.

Algorithm 3 presents the FedSCA conceptual algorithm according to the weight update in Eq 6. Based on Algorithm 2, the algorithm is expanded using SCA. In contrast to typical methods, the Function CLOUDSERVEREXECUTES receives just the best values and does not request w from the clients in Line 6. Lines 7 and 8 carry out the process of identifying the client with the minimal best value among those obtained. Function LocalClientUpdate proceeds the ANN applying the SCA. $C_{X_{itr}}$ is calculated in lines 17-19 and supplied to the server. For each epoch i, repeat the iterations (training) from lines 20 to 23. Function The GetBestModel method (lines

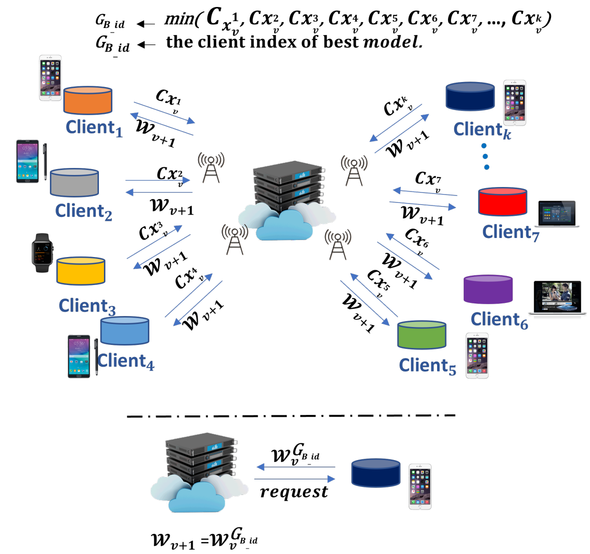

Fig. 3. The process of updating the weights of the FedSCA algorithm. The client with the best score value receives a request from the server to be used as the global model after the server receives the scores from all clients.

25–28) asks the server for the model with the best score. The code is available in the FedSCA GitHub [17].

Algorithm 3: Pseudocode of the FedSCA algorithm

```
1: Set the SCA parameters (a, c_1, Number of dimensions, Number of solution,
   Number of iterations Max_itr).
2: Function CloudServerExecuts :
3: Initialize global model (w_0), G_B_i^itr, G_B_id.
4: for each iteration itr = 1, 2, 3...Max_itr do
5:     for each client k do
6:         G_B_id=LocalClientUpdate(k,G_B_id)
7:         Update G_B_i^itr
8:         Update G_B_id
9:     end for
10:    w^{t+1}=GetBestModel(G_B_id)
11: end for
12: Function LocalClientUpdate(k,G_B_id) :
13: Initialize Cx_itr, a, c_1.
14: B= split data into batches.
15: Compute c_1 using Eq (4).
16: Generate c_2, c_3, and c_4.
17: for each layer l=1,2,3... do
18:     Update the current solution using Eq (6).
19: end for
20: for each client epoch i=1,2,3...E do
21:     for batch b ∈ B do
22:         Update w.
23:     end for
24: end for
25: Return Cx_itr.
26: Function GetBestModel((G_B_id):
27: Request to Client(G_B_id)
28: Receive w from Client
29: Return w to server.
```

IV. Experiments

We tested FedSCA's accuracy and convergence speed, as well as conducted tests in an Unstable Network Environment (UNE), to determine its effectiveness. We wanted to see whether the FedSCA had sufficient convergence speed and accuracy, given its smaller amount of network communication than FedAvg. Furthermore, we compared the proposed algorithm with another population-based optimization algorithm (i.e., FedPSO). On the other hand, we looked at the client-server data connection costs using the CIFAR-10 dataset [18] as an accuracy criterion for the three techniques. FedAvg, FedPSO, and FedSCA were put to the test under different network conditions to determine their accuracy.

A. Experimental Setup

A CPU of 2.50 GHz (2 processors) with 128 gigabytes (128 gigabytes usable), Xeon(R), an Intel(R) and 465 gigabytes of memory was used for the experiments. Experiments were conducted using TensorFlow 2.3.0 and Keras 2.4.3.

A new data format was sent from the client to the server, and the distributed model's weights were altered as a result. The CNN model architecture was taken from [10] (the first with 32 channels, the second with 64 channels, each followed by 2 x 2 maximum pooling). Table I provides a breakdown of the model's layers.

TABLE I
CNN PARAMETER SETTINGS

ID	Shape	Layer
1	5 x 5 x 32	Conv2D
2	32	Conv2D
3	5 x 5 x 64	Conv2D
4	64	Conv2D
5	1024 x 512	Dense
6	512	Dense
7	512 x 10	Dense
8	10	Dense

The experiment was carried out using the CIFAR-10 dataset. For image classification, CIFAR-10 is a popular dataset. Training and test images are included, with 32 x 32-pixel images from ten different categories, such as automobiles and planes. The CIFAR-10 dataset was shuffled, assigned to K clients, and distributed to each client to proceed with training. With the exception of the dropout layer, no independent tuning method was applied to increase the accuracy during training. FedSCA and FedAvg client training had a learning rate of 0.0025. Table II displays the hyperparameter values that were employed in this work.

TABLE II
EXPERIMENT HYPERPARAMETERS

ID	Parameter	FedSCA	FedAvg
1	Batch	10	10
2	Client-epoch	5	5
3	Epoch	30	30
4	C	-	0.1, 0.2, 0.5, 1.0
5	Client	10	10

B. Experimental Results for Accuracy

Experimental results for accuracy using the CIFAR-10 dataset are depicted in Figure 4 and Table III. For curve, we used a test accuracy as our basis. FedSCA's accuracy was better (72.41%) than FedAvg's at all 30 epochs, and it was superior from the start. C = 1.0 was FedAvg's most accurate setting, with a best accuracy of 67%. In FedAvg training, C is a constant between 0 and 1, which restricts the number of clients used. Clients who scored at least C in each communication round were chosen from the rest of the group. As the value of C increases in Figures 4 and 5, the more accurate data is transferred between server and client, the larger the data size transmitted. At C = 0.5, where data transfer costs are equivalent, the accuracy disparity widens (65.00% for FedSCA).

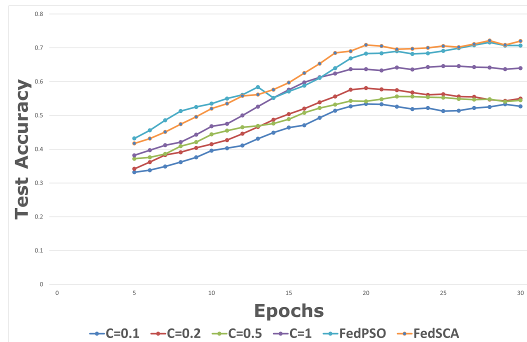

Fig. 4. A comparison of the test accuracy of different algorithms.

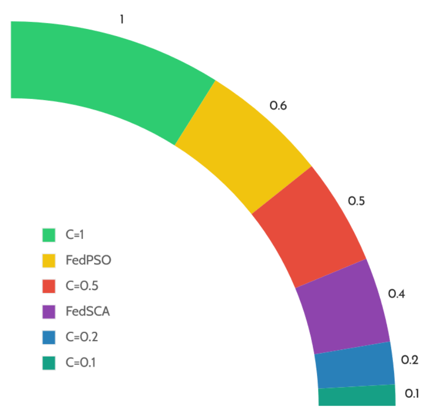

Fig. 5. A comparison of the communication cost of different algorithms.

TABLE III
TEST ACCURACY COMPARISONS

Algorithm	Accuracy (Testing)
FedAvg,C = 0.1	51.39%
FedAvg,C = 0.2	59.07%
FedAvg,C = 0.5	65.00%
FedAvg,C = 1.0	67.14%
FedPSO	70.12%
FedSCA	**72.41%**

C. Experimental Results for Unstable Network Environments

We then simulated an UNE. In each communication cycle, random data transmissions from client to server were lost. We used data discarded in ranges of 0%, 10%, 20%, and 50% to validate the accuracy discrepancy between the two methods. In order to ensure the experiments' validity, all 10 trials were done until the average value was reached. Table IV depicts the outcome of discarding data at random for FedAvg when C = 1.0, FedPSO, and FedSCA. FedAvg demonstrates an average accuracy reduction of 6.43 % due to random data dropouts. The same table displays the findings for FedSCA, which had a 2.43 % decline in average accuracy. In an experiment evaluating the model in an UNE where the data cannot be delivered in its whole, FedSCA's accuracy reduction was 4 % better than FedAvg.

TABLE IV
ACCURACY AGAINST COMMUNICATION FAILURE PROBABILITY.

Algorithm	Failure Rate			
	50%	20%	10%	0%
FedAvg,C = 1.0	59.55%	61.09%	61.48%	67.14%
FedPSO	65.47%	68.41%	69.18%	70.12%
FedSCA	**66.37%**	**68.37%**	**71.39%**	**72.41%**

V. CONCLUSION

In this paper, the Sine Cosine Algorithm (SCA) is used to improve the FL's network communication performance by utilizing the SCA mechanism to update the weights of learned models as well as reducing the amount of data being sent from clients to servers. The server-trained model's score value is distributed in the proposed method, which aggregates the results. The server receives the trained model from the highest-scoring client. A two-layer Convolutional Neural Network (CNN) is created to test the proposed algorithm on the CIFAR-10 dataset. The proposed FedSCA algorithm had an accuracy improvement of 9.87% and 2.29% on average than FedAvg and FedPSO, respectively. Additionally, FedSCA is on average 4% more resilient than FedAvg and FedPSO when the communication data is randomly dropped. We want to use a range of SCA implementations in the future in order to improve network connection. It is possible to avoid local minima by using dynamic multiple swarm SCA and P2P-SCA client communication. We plan to use a range of network protocols, including the gossip protocol, due to the high frequency of client drop-offs and the restricted network capacity. This is in addition to the fact that when the ANN layer count increases, the size of the model also increases. Eventually, we will conduct an experiment to see if the results can be replicated in a model with deeper layers.

ACKNOWLEDGEMENTS

This work was supported by the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), UAE under grants No: 8481000015, 8481000021.

REFERENCES

[1] M. Milicevic and S. Eybers, "The challenges of data analytics implementations: A preliminary literature review," in *Proceedings of International Conference on Data Science and Applications*. Springer, 2022, pp. 27–36.

[2] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[3] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information Processing Management*, vol. 59, no. 6, p. 103061, 2022.

[4] S. Park, Y. Suh, and J. Lee, "Fedpso: federated learning using particle swarm optimization to reduce communication costs," *Sensors*, vol. 21, no. 2, p. 600, 2021.

[5] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[6] M. Aloqaily, I. Al Ridhawi, and M. Guizani, "Energy-aware blockchain and federated learning-supported vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[7] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "Fedco: A federated learning controller for content management in multi-party edge systems," in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–9.

[8] S. Mirjalili, "Sca: a sine cosine algorithm for solving optimization problems," *Knowledge-based systems*, vol. 96, pp. 120–133, 2016.

[9] G. Wainer, M. Aloqaily *et al.*, "Machine learning-based indoor localization and occupancy estimation using 5g ultra-dense networks," *Simulation Modelling Practice and Theory*, vol. 118, p. 102543, 2022.

[10] 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.

[11] V. Balasubramanian, M. Aloqaily, M. Reisslein, and A. Scaglione, "Intelligent resource management at the edge for ubiquitous iot: an sdn-based federated learning approach," *IEEE network*, vol. 35, no. 5, pp. 114–121, 2021.

[12] O. Bouachir, M. Aloqaily, Ö. Özkasap, and F. Ali, "Federatedgrids: Federated learning and blockchain-assisted p2p energy sharing," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 424–436, 2022.

[13] Z. A. A. Alyasseri, O. A. Alomari, S. N. Makhadmeh, S. Mirjalili, M. A. Al-Betar, S. Abdullah, N. S. Ali, J. P. Papa, D. Rodrigues, and A. K. Abasi, "Eeg channel selection for person identification using binary grey wolf optimizer," *IEEE Access*, vol. 10, pp. 10 500–10 513, 2022.

[14] A. K. Abasi, A. T. Khader, M. A. Al-Betar, S. Naim, S. N. Makhadmeh, and Z. A. A. Alyasseri, "An improved text feature selection for clustering using binary grey wolf optimizer," in *Proceedings of the 11th national technical seminar on unmanned system technology 2019*. Springer, 2021, pp. 503–516.

[15] O. A. Alomari, S. N. Makhadmeh, M. A. Al-Betar, Z. A. A. Alyasseri, I. A. Doush, A. K. Abasi, M. A. Awadallah, and R. A. Zitar, "Gene selection for microarray data classification based on gray wolf optimizer enhanced with triz-inspired operators," *Knowledge-Based Systems*, vol. 223, p. 107034, 2021.

[16] A. K. Abasi, A. T. Khader, M. A. Al-Betar, S. Naim, S. N. Makhadmeh, and Z. A. A. Alyasseri, "A novel ensemble statistical topic extraction method for scientific publications based on optimization clustering," *Multimedia Tools and Applications*, vol. 80, no. 1, pp. 37–82, 2021.

[17] A. K. Abasi, M. Aloqaily, and M. Guizani, 2022. [Online]. Available: https://github.com/Artifitialleap-MBZUAI/Sine-Cosine-Algorithm-for-Reducing-Communication-Costs-of-Federated-Learning

[18] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html