ASYNCHRONOUS FEDERATED LEARNING: CONVERGENCE AND PERFORMANCE IN HETEROGENEOUS ENVIRONMENTS

Dan Gabriel BADEA[1*], Ștefan-Dan CIOCÎRLAN[2], Răzvan-Victor RUGHINIȘ[3], Dinu ȚURCANU[4]

Distributed Machine Learning (DML) utilizes several nodes for training machine learning models, with a central node overseeing data distribution and communication. Federated Learning (FL), a Distributed Machine Learning (DML) branch, improves data privacy by retaining data on local devices. The realm of Federated Learning (FL) has primarily functioned synchronously; nevertheless, recent advancements have initiated a new phase of asynchronous FL. This innovative method enables nodes to update the model, independently facilitating exceptional scalability and adaptability. This study thoroughly examines asynchronous federated learning, analyzing its distributed architecture, communication protocols, optimization methods, and the numerous hurdles it faces, such as data heterogeneity, node delays, and convergence problems. The findings illustrate that, despite these challenges, the system attains near-centralized accuracy and exhibits accelerated convergence rates. This serves as a compelling demonstration of the potential of asynchronous federated learning in transforming actual applications.

Keywords: FL - Federated Learning, DML - Distributed Machine Learning, Convergence

1. Introduction

This research focuses on Distributed Machine Learning (DML), a technique for training machine learning models across several nodes. This approach helps overcome the memory limitations often encountered with systems that rely on a single node [1]. Data is aggregated at a central server and distributed to nodes, improving the model's performance, accuracy, and scalability for large datasets.

[1*] Eng., National University of Science and Technology POLITEHNICA Bucharest, Romania, *corresponding author, e-mail: dan_gabriel.badea@upb.ro
[2] Lecturer, PhD Eng., National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: stefan_dan.ciocirlan@upb.ro
[3] Professor, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: razvan.rughinis@upb.ro
[4] Professor, Technical University of Moldova, Republic of Moldova, e-mail: dinu.turcanu@adm.utm.md

Federated Learning (FL) has revolutionized DML by allowing distributed training across various devices while prioritizing data privacy [3]. Unlike DML, where data is transferred to a central location for training, FL maintains data on local devices, transmitting only model updates to a central server. Synchronous FL, where nodes wait for each other to complete their updates, has been widely used but presents challenges in real-world applications, especially in heterogeneous environments with inconsistent network availability [15].

In federated learning (FL), the challenges associated with synchronized updates have prompted the emergence of asynchronous FL as a viable solution, offering enhanced system flexibility and scalability [16]. Nonetheless, asynchronous FL presents unique challenges, including slower convergence rates resulting from inconsistent data distributions and divergent computational capacities among devices [16]. Consequently, these factors can lead to delayed model improvements and suboptimal performance. To address these issues, this study proposes integrating adaptive weighting strategies into the aggregation process of asynchronous FL. This strategic approach aims to mitigate the challenges associated with asynchronous updates, ultimately enhancing the efficacy of the learning process. Instead of treating all updates from participants equally, the system dynamically adjusts the weights based on the quality, relevance, and timeliness of the data provided by each participant. This ensures that contributions from devices with more informative data are prioritized, leading to faster model convergence and improved accuracy [3]. This approach also helps mitigate the impact of delayed or inconsistent updates from slower participants, a common problem in both synchronous and asynchronous setups [2]. (selected to assess scalability and convergence behavior under different load conditions) (particularly important for reducing computational load in resource-limited environments such as IoT)

This research addresses the primary challenge of reducing computational time in a distributed system while maintaining high model accuracy. Traditional DML architectures face synchronization bottlenecks, as the slowest devices can delay the entire training process. Moreover, centralized servers expose a vulnerability where third parties can access participant data, undermining privacy guarantees [2].

By penalizing clients that transmit updates infrequently, this research proposes a system that maintains robust performance even with infrequent updates, further ensuring security and stability against potential malicious participants [2]. This study also extensively evaluates adaptive weighting in asynchronous FL through experimental analysis of real-world datasets. Different weighting strategies were applied to participant updates, considering the distribution and quality of data. Results show that adaptive weighting reduces training time and improves convergence speed compared to traditional synchronous FL methods, especially in

scenarios with data heterogeneity and intermittent device availability [9]. (selected to assess scalability and convergence behavior under different load conditions)

A foundational Federated Learning (FL) model was fine-tuned during the experimental evaluation by incorporating an age parameter and precision metrics. The research involved a detailed comparison between centralized and distributed models, examining their computational time, energy consumption, and convergence rate. The **Street View House Numbers (SVHN)** dataset was chosen due to its size, with varying numbers of participants (5, 10, 15) and iterations (50, 100) tested. The distributed model achieved an accuracy of **88.71%**, closely mirroring the centralized model's performance of **90.54%**, with faster convergence due to the asynchronous implementation [2]. (selected to assess scalability and convergence behavior under different load conditions)

The comprehensive research findings highlight that integrating adaptive weighting strategies, such as differential privacy and personalized local updates, into asynchronous FL significantly accelerates convergence and substantially enhances model performance in distributed environments. By leveraging techniques such as personalized local updates, where each client device updates its local model independently, the study demonstrates a marked improvement in the overall performance and privacy preservation of federated learning models. These results contribute significantly to the ongoing development of federated learning technologies and underscore the importance of optimizing the use of distributed data while maintaining high levels of privacy and performance. This groundbreaking research provides a solid foundation for further investigation into the practical applications of asynchronous federated learning across various real-world settings, including healthcare, finance, and IoT networks. (particularly important for reducing computational load in resource-limited environments such as IoT)

2. Literature Review

Distributed Machine Learning (DML) provides valuable insights into optimizing the training of machine learning models across multiple nodes, effectively addressing the memory limitations associated with single-node architectures. One proposed approach involves utilizing personal devices as computational nodes, enabling the processing of local user data without overburdening a central data center. In this system, the central node primarily serves as a coordinator, decoupling data from the central node to minimize communication frequency, which is proposed to occur once daily. Model updates, represented as small vectors of model weights, are centrally aggregated based on data distribution, ensuring computational efficiency even in the absence of independently and identically distributed (IID) datasets. The introduction of a novel algorithm

demonstrates promising experimental results, particularly for convex problems in distributed environments, with Federated SVRG (Stochastic Variance Reduced Gradient) improving upon traditional methods such as Gradient Descent and CoCoA (Communication-efficient distributed optimization with composite oracles), achieving satisfactory error rates with fewer communication rounds, albeit requiring further enhancements.

In a separate study, authors propose strategies to mitigate communication costs between servers and clients when transmitting models. They argue that upload times typically exceed download times, suggesting that structured updates, which involve learning from a limited parameter space with fewer variables, can alleviate this bottleneck. Evaluations across multiple datasets demonstrate that client selection can be expanded without compromising accuracy, particularly effective in bandwidth-constrained environments. A vital outcome of this approach is the achieved balance, allowing for the inclusion of more clients per communication round. At the same time, each transmits less data, ensuring scalability, especially in scenarios where clients have limited bandwidth, a common real-world constraint.

Experiments detailed in a different study showcase the robustness of the Federated Averaging (FedAvg) algorithm in addressing non-IID and imbalanced data distributions, common characteristics of real-world datasets. By significantly reducing communication rounds compared to synchronized stochastic gradient descent (SGD), FedAvg emerges as a practical solution for real-world federated learning systems. FedAvg integrates local SGD on each client with central aggregation, leading to substantial resource savings while maintaining user privacy. The study's findings demonstrate that federated learning significantly improves resource consumption without compromising accuracy, indicating the algorithm's effectiveness across various model architectures, including multi-layer perceptrons and word-level LSTM models.

In another development, FedProx, introduced in a separate study, extends the capabilities of FedAvg by addressing the interaction between system heterogeneity and statistical heterogeneity. The study emphasizes how the exclusion of clients due to system constraints can exacerbate statistical heterogeneity. FedProx addresses this by permitting partial work on devices and incorporating a proximal term to aggregate partial updates safely. This modification presents a more reliable approach to handling diverse client data without compromising performance. The study comprehensively explores these modifications, highlighting their significant implications for real-world federated systems where heterogeneity is an inevitable challenge.

Furthermore, a study presents a scalable Federated Learning (FL) production system on mobile devices developed using TensorFlow. This work addresses ensuring sufficient devices connect simultaneously to maintain task progress and uphold security properties. The proposed probabilistic algorithm for

device reconnection reduces the need for additional device-server communications, ensuring operational efficiency in live environments. The system also monitors the device health metrics, including memory usage, error rates, and battery consumption, to prevent resource wastage. This work marks one of the initial large-scale implementations of FL in production settings, explicitly focusing on federated mediation algorithms operating directly on mobile devices, indicating broad implications for future FL deployments beyond mobile settings.

A separate article demonstrates the efficacy of using federated averaging to train recurrent neural network (RNN) models on client devices compared to traditional server-based methods. The study focuses on the Gboard system's word prediction, illustrating how federated learning maintains data privacy by preserving user data on devices without transferring it to central servers. Unlike server-based learning, which necessitates large uniformly distributed datasets across servers, federated learning aggregates model updates while safeguarding user privacy. This technique has been successfully deployed on Gboard, enhancing suggestions for hundreds of millions of users in 2021.

3. Architecture

The architectural solution for this project involves using a central server to manage the primary model and coordinate with other participants (workers). This approach optimizes energy consumption at the expense of potentially lower model performance than training on a single dataset. Each client registers with the central server to receive a pre-trained model for optimal initialization. With the server aggregating participant updates, both trusted clients and potential adversaries are considered. The asynchronous approach ensures real-time model updates as clients send new data, allowing workers to receive the updated model through weight sharing. The model is trained in each iteration using local data, and the updates are transmitted to the central node. The central server aggregates the updates from all active participants and adjusts the general model’s weights accordingly. The updated model is then shared with the participants by only transmitting the modified weights. Clients with delayed responses have a reduced impact on the central node’s updates. The asynchronous mechanism updates the central model for each client’s new information and notifies other participants when they transmit their data. Upon receiving the model, each participant independently trains it using their local dataset, which may consist of diverse data points with varying characteristics, such as size, distribution, and quality. The learned values from these individual training processes are combined through weighted averaging to improve the overall model. It's crucial to account for the data heterogeneity present in participant datasets, as differences in volume and distribution can significantly impact the convergence and generalization of the final model. The importance of each participant's contribution

is carefully assessed using the functions proposed in this work. This assessment reveals that a significant majority, estimated to be over 80% of the model's weight is attributed to key contributors. The determination of participant weights is based on the functions (6), (7), (8), and (9) outlined in this paper, which consider the unique characteristics of each participant's dataset and the impact of their contribution on the overall model's performance. This proposed architecture ensures the system can converge toward a well-performing, generalized model even with heterogeneity among participants and asynchronous updates. (selected to assess scalability and convergence behavior under different load conditions)

$$N = number\ of\ participants$$
$$G = age\ of\ the\ central\ server's\ general\ model$$
$$W = age\ of\ the\ current\ participant's\ model$$
$$f_r = frequency\ of\ consecutive\ participation\ by\ the\ client$$
$$acc = most\ recent\ accuracy$$

$$f_1(\mathrm{N, G, W}, f_r, \mathrm{acc}) = \frac{1}{(acc \cdot f_r \cdot (N + G - W))}$$

$$f_2(\mathrm{N, G, W}, f_r, \mathrm{acc}) = \frac{acc}{(N \cdot f_r)} \cdot \frac{1}{(G + 1 - W)}$$

$$f_3(\mathrm{G, W}, f_r, \mathrm{acc}) = \frac{1}{(acc \cdot f_r \cdot \frac{(2 \cdot G + 1 \cdot W)}{3})}$$

$$f_4(\mathrm{N, G, W}, f_r, \mathrm{acc}) = \frac{(f_1 \cdot f_2)}{f_r}$$

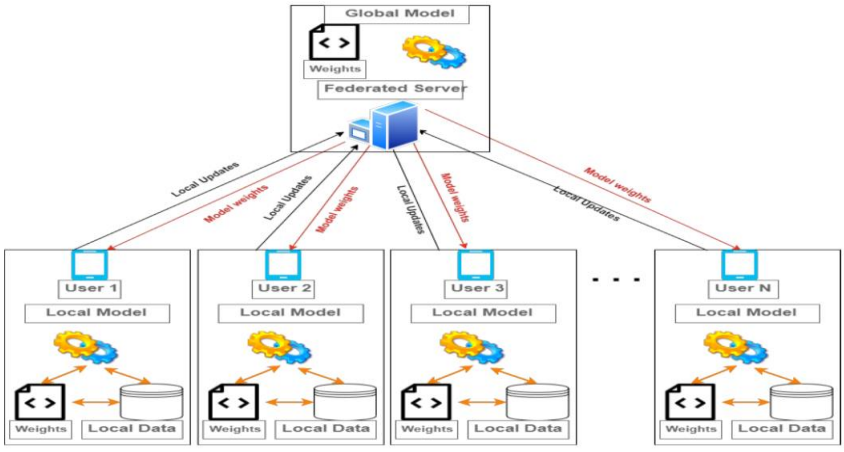

Fig. 1. General architecture

The age of each participant (W) or the general model (G) represents the duration of time or synchronization cycles since the last update between the participant's model and the central model. This age difference directly impacts the weight assigned to the participant's contribution during the aggregation process, with more significant discrepancies leading to a reduced influence, thereby penalizing outdated models. The frequency parameter (fr) further accounts for the frequency of a participant's appearance in random aggregation, penalizing repeated appearances by diminishing the potential for new improvements over consecutive iterations. In addition, the accuracy (acc) parameter plays a crucial role in the weight distribution, ensuring that models deviating significantly from the general model are proportionally assigned less influence during the aggregation process. (selected to assess scalability and convergence behavior under different load conditions).

Functions (f_1), (f_2), (f_3), and (f_4) determine the weight distribution based on parameters (N), (G), (W), (f_r), and (acc), balancing the most promising and least-performing updates. The general model's weight importance is given by:

$$1 - avg_{weight}$$

Where avg_{weight} results from applying any of the functions f_1, f_2, f_3 or f_4.

4. Evaluation

The application was developed and run locally on a computer with the following specifications:

- Intel i5-10500F
- 32GB DDR4 2933MHz RAM
- QUADRO RTX 4000 (12GB VRAM)
- 256GB SSD

The system architecture incorporates a specialized class to store participant-specific information, encompassing details such as the participant's name, the time since the last model update, the dataset used for training, the frequency of consecutive updates, and the most recent accuracy. Initially, data distribution among participants can be equal or unequal. A list of participants is generated, and a random participant is selected for each iteration, thereby simulating an asynchronous environment. The chosen participant proceeds to train on their local dataset, calculating the importance of weight for aggregation with the general model's weights. (selected to assess scalability and convergence behavior under different load conditions)

The central model is systematically updated with each participant's contribution, and this process continues for a predetermined number of iterations.

An extensive analysis of various parameters was conducted, including the number of iterations (50, 55, 100), different functions to calculate the participant's weight importance, and varying numbers of participants (5, 10, 15)—additionally, a thorough evaluation of the training time before aggregation was carried out. (selected to assess scalability and convergence behavior under different load conditions)

Two distinct scenarios were rigorously tested: one involving an even distribution of data and another featuring unequal distribution to simulate real-world conditions. Furthermore, the system's resilience against malicious participants was carefully evaluated. Specific models were intentionally introduced with irrelevant or random weights to assess their impact on the overall model's performance. This was done by initializing specific participants with zero or random weights to gauge their influence. (selected to assess scalability and convergence behavior under different load conditions)

Regarding data exposure, it was assumed that the data was homogeneous, disregarding differences between clients. Figure 2 provides a detailed demonstration of the model's performance over a centralized 60-epoch training period. The model successfully achieved convergence in 30 epochs, stabilized at 50 epochs, and reached a final accuracy of 88.94%. This accuracy threshold was established, as further training would not significantly enhance the test accuracy beyond 90%.

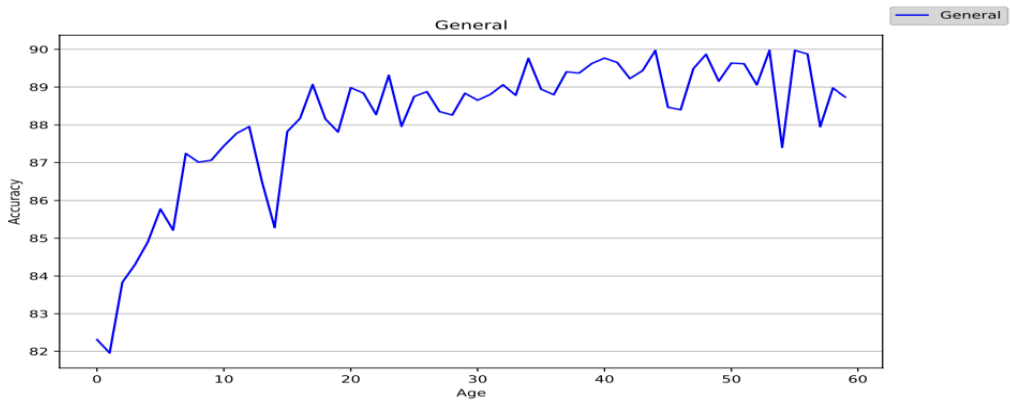

Fig. 2. Centralize model

4.1 Subsection (A)

This subsection examines the general model's performance using the client weight importance function f_1 (6). The figures illustrate that the model converges rapidly, stabilizing around the 20th epoch. Accuracy fluctuations are minimal, and the final results closely align with the centralized model's performance when participants act honestly. However, when a participant initializes model weights to

zero, the f_1 function becomes more sensitive, highlighting its responsiveness to poorly performing or intentionally flawed models in the federated learning process. (selected to assess scalability and convergence behavior under different load conditions)

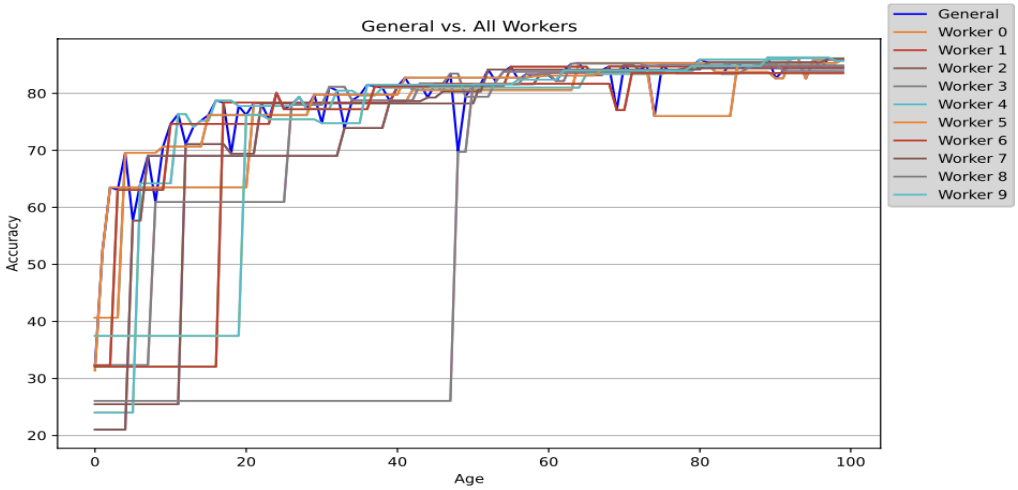

Fig. 3. General model & Workers model accuracy evolution using weight function $\boldsymbol{f_1}$

4.2 Subsection (B)

In this subsection, the general model's evolution is examined using the client weight importance function $\boldsymbol{f_2}$ (7). The figures show that the model converges faster, stabilizing around the 15th epoch with fewer steps. Accuracy fluctuations are kept within a 3% margin, and the final results show only a 2% drop in accuracy compared to the centralized model.

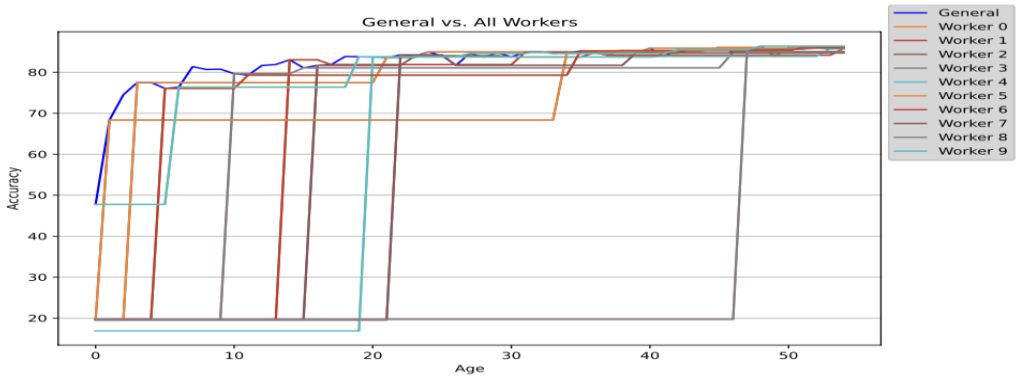

Fig. 4. General model & Workers model accuracy evolution using weight function $\boldsymbol{f_2}$

Malicious participants have a significantly reduced impact on the overall mediation process, both on the general model and other participants, though their

influence on the final model quality remains slightly noticeable. (selected to assess scalability and convergence behavior under different load conditions)

4.3 Subsection (C)

This subsection presents the results obtained using the client weight importance function f_3 (8). The figures show that the general model struggles with convergence, stabilizing slowly and remaining volatile. Accuracy fluctuations exceed a 10% margin, and the final results indicate an 11% decrease in overall accuracy compared to the centralized model. Malicious participants exacerbate the instability, leading to the weakest outcome observed using this weighting function. Their presence significantly undermines the model's performance, demonstrating the function's limitations in handling adversarial inputs. (selected to assess scalability and convergence behavior under different load conditions)

Fig. 5. General model & Workers model accuracy evolution using weight function f_3

4.4 Subsection (D)

In this subsection, we analyze the performance of the general model using the client weight importance function f_4 (9). The graphs show that the model converges quickly, requiring fewer iterations to stabilize around the 20th epoch. Accuracy fluctuations remain within a 4% margin, and the final results show only a 2% reduction in accuracy compared to the centralized model. Malicious participants have a diminished impact on the general model's performance and the other participants. The use of this function, while less effective than f_2, shows that hybrid approaches can be equally efficient. (selected to assess scalability and convergence behavior under different load conditions)

Fig. 6. General model & Workers model accuracy evolution using weight function f_4

5. Conclusions

The findings of this research demonstrate that asynchronous distributed learning can be successfully implemented, with minimal performance differences between centralized and decentralized models, provided that data privacy remains a priority. Preliminary results indicate that the decentralized general model can capture essential features within a short training period. Utilizing the weight importance functions f_2 (7) and f_4 (9) has proven to be an efficient and secure method for federated learning. These functions offer substantial advantages, including enhanced protection against malicious users and faster convergence, while safeguarding data privacy. By applying f_2 and f_4 will effectively manage malicious participants by detecting and mitigating their impact. This ensures minimal model degradation or corruption risk, thus improving system security and integrity during distributed learning. (selected to assess scalability and convergence behavior under different load conditions)

In addition to protecting against adversarial actors, these functions enable faster convergence, achieving model stability and desired performance in fewer steps. This results in better and quicker outcomes within the federated learning framework. Another key benefit of the approach using f_2 and f_4 is the preservation of data privacy. By transmitting only model weights, rather than raw data, the approach ensures that sensitive user information remains protected, minimizing exposure risks during the aggregation process.

In conclusion, the use of f_2 and f_4 in federated learning is both effective and secure, providing multiple benefits in terms of protection against malicious users, faster model convergence, and data privacy preservation. This approach performs comparably to traditional methods while offering additional advantages for

managing adversarial users and enhancing the efficiency of the federated learning system.

R E F E R E N C E S

[1] McMahan, and Brendan. “Communication-efficient learning of deep networks from decentralized data.” Artificial intelligence and statistics, vol. PMLR, 2017.
[2] Konečný, Jakub, et al. "Federated learning: Strategies for improving communication efficiency." NIPS Workshop on Private Multi-Party Machine Learning. 2016
[3] Wang, J., et al. "Advances in federated learning: A survey." IEEE Transactions on Big Data (2021)
[4] Konecny, and Jakub. “Federated optimization: Distributed machine learning for on-device intelligence.” 2016
[5] Li Li, et al. “A review of applications in federated learning.”
[6] Peter Kairouz, et al. “Advances and Open Problems in Federated Learning.”
[7] Qiang Yang, et al. ”Federated Learning”.
[8] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19
[9] Tian Li, et al. “Federated Learning Challenges, methods, and future directions.”
[10] McMahan, Brendan, and Daniel Ramage. “Federated Learning: Collaborative MachineLearning without Centralized Training Data.” Google AI Blog, 6 April 2017
[11] Keith Bonawitz, et al. “Practical Secure Aggregation for Privacy-Preserving Machine Learning.” Cornell Tech, 2 West Loop Rd., New York, NY 10044
[12] Keith Bonawitz, et al. “TOWARDS FEDERATED LEARNING AT SCALE: SYSTEM DESIGN”.
[13] Hard, Andrew, et al. "Federated learning for mobile keyboard prediction." Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018
[14] Li, Tian, et al. "FedAsync: Asynchronous federated learning for distributed deep learning." arXiv preprint arXiv:1902.02089 (2019)
[15] Hsieh, K., et al. "A survey on federated learning: Challenges, advances, and future directions." IEEE Transactions on Big Data (2020)
[16] Hsieh, K., et al. "A survey on federated learning: Challenges, advances, and future directions." IEEE Transactions on Big Data (2020)
[17] Li, Tian, et al. "Federated optimization in heterogeneous networks." arXiv preprint arXiv:1812.06127 (2018)