Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 871–876
DOI: 10.15439/2023F722
ISSN 2300-5963 ACSIS, Vol. 35

Diagnosing Machine Learning Problems in Federated Learning Systems: A Case Study

Karolina Bogacka, Anastasiya Danilenka
0000-0002-7109-891X, 0000-0002-3080-0303
Warsaw University of Technology
Plac Politechniki 1, 00-661 Warszawa, Poland
Email: {karolina.bogacka, anastasiya.danilenka}.dokt@pw.edu.pl

Katarzyna Wasielewska-Michniewska
0000-0002-3763-2373
Systems Research Institute, Polish Academy of Sciences
Newelska 6, 01-447 Warszawa, Poland
Email: katarzyna.wasielewska@ibspan.waw.pl

***Abstract*—The proliferation of digital artifacts with various computing capabilities, along with the emergence of edge computing, offers new possibilities for the development of Machine Learning solutions. These new possibilities have led to the popularity of Federated Learning (FL). While there are many existing works focusing on various aspects of the FL process, the issue of the effective problem diagnosis in FL systems remains largely unexplored. In this work, we have set out to artificially simulate the training process of four selected approaches to FL topology and compare their resulting performance. After noticing concerning disturbances throughout their training process, we have successfully identified their source as the problem of exploding gradients. We have then made modifications to the model structure and analyzed the new results. Finally, we have proposed continuous monitoring of the FL training process through the local computation of a selected metric.**

I. Introduction

FEDERATED Learning (FL, [1], [2]) is a relatively novel approach to Distributed Machine Learning (DML). It allows a system to take full advantage of the data locality and computing power of distributed devices. In a standard scenario, the goal is to train a global model using local models trained by the clients on their local data. Clients periodically send parameter updates to an aggregator node. The new version of the global model is established, communicated back to clients and the process repeats until stopping criteria are met.

Currently, a common way to prepare for an FL training process begins with the centralized construction and training of an initial ML model. This preliminary phase allows the developer to utilize a plethora of already established techniques in order to develop the best ML solution possible. The data preprocessing steps, architecture and hyperparameters from that solution are then used as a basis for the local models trained by the FL clients. However, this approach also relies on the existence of a global dataset with a distribution and format that sufficiently resembles that of the client data. This global dataset may sometimes be impossible to create due to the client data being very localized, client-specific and inaccessible because of privacy concerns. Of course, such a model can also be developed as an FL model from scratch by conducting multiple FL training runs and selecting the best performing training parameters and architectures. Unfortunately, the diagnosis of problems such as vanishing or exploding gradients based on the learning curve would necessarily be hindered by the existence of other destructive factors, like client dropout and differing local data distributions. The development of the final model would therefore necessitate a large number of completed FL training processes and as such be both very resource and time consuming.

Additionally, many current research trends in FL result in solutions that may undergo a very different training process from the centralized baseline. For example, a common goal of trying to achieve scalability while preserving the stability of the training is often mitigated through the appropriate choice of topology. Here, topology refers to the network topology of the FL system, which indicates how clients communicate with the server and with each other [3]. The additional communication on the global and local level may cause the training curve to undergo periodic spikes and drops in accuracy, which can then be hard to distinguish from other ML problems.

We have encountered the problems mentioned above throughout our work on the Assist-IoT project [1]. We were conducting tests in order to select the FL topology best suited to use in the Assist-IoT project in the pilots focusing on: (1) construction workers' health and safety assurance, (2) vehicle exterior condition inspection [4]. The purpose of the FL solution was effective fall detection of construction workers in the case of (1) and automatic vehicle damage detection in the case of (2). In an effort to determine the best topology for the aforementioned use cases, we have analyzed different approaches to the problem [5] and selected 4 most "promising" and representative to further test their behaviour. Our experiments have revealed concerning instabilities in the training processes of some of them. Through additional trials and further examination of the existing result, we have identified the source of the problem as exploding gradients. The problem of exploding gradients here describes a situation in which the gradient backpropagated through a neural network grows exponentially during training, causing the neural network performance to stall or even deteriorate [6].

The work of Karolina Bogacka and Anastasiya Danilenka was funded in part by the Centre for Priority Research Area Artificial Intelligence and Robotics of Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme.

[1] https://assist-iot.eu

We have then modified the hyperparameters to avoid this problem and achieve better results. We would like to share our process as a case study, finishing it with a proposal of an additional procedure that could enable easier identification of the exploding gradient problem in FL systems.

II. Related works

A. The state of FL diagnostic tools

There are few existing works concerning the design and development of FL diagnostic tools. Some of them focus on system monitoring through the identification of badly-performing clients while maintaining security and privacy [7] [8]. This approach to diagnostics is appropriate for FL systems, however, it is often unable to sufficiently recognize any problems stemming from the model architecture or training configuration. The most comprehensive attempt at providing systematic problem diagnosis for FL, FedDebug, offers the possibility of setting breakpoints and replaying previous rounds through a continuous collection of metrics, such as response time, training and validation loss as well as other performance indicators. Those metrics are then used to construct a simulation of the training, which enables easier identification of a faulty client. Despite the narrow focus of this solution, the broad functionalities of the system should be easily extended to other problems [9].

One of the rare works to not focus on finding underperforming clients but on mitigating issues concerning FL models is Fed-DNN-Debugger [10]. It consists of two modules: one is responsible for nonintrusive metadata capture (NIMC), which produces data that is then used for automated neural network model debugging (ANNMD). Local models are then repaired through retraining on specially selected samples. However, its main focus lies in training bugs, which are caused by misconducted training processes, such as biased data, noisy data or insufficient training. It does not enable easy identification of structure bugs, which stem from inappropriate model architecture or hyperparameters.

B. The exploding gradient problem

The exploding gradient problem describes a phenomenon in which the gradient backpropagated through a neural network grows exponentially from layer to layer [6]. Unfortunately, the maximal depth of many popular ML architectures is limited by the existence of this phenomenon. There are existing techniques such as weight scaling or batch normalization which can be used to mitigate these problems. However, they are not always effective [6]. It is possible to use architectures that avoid the exploding gradient problem [11] such as fully connected ReLU networks. Nevertheless, due to the limited functionality of those architectures, it is not a commonly employed practice.

C. Advances in research on Topology of Federated Learning

Although a typical FL system follows a simple centralized topology, with a single server node, often located in the cloud, communicating directly with a federation of clients, this is not necessarily the most optimal, or efficient, solution for many use cases [3]. Interest in network topology in the context of FL stems from the evidence that its impact can be extremely effective in mitigating data heterogeneity. Some types of topologies can also either fully eliminate the need for a central cloud server or greatly reduce its importance [12]. This is significant since the main roadblock for the full production deployment of many FL systems involves communication inefficiency. Other works try to balance these two approaches, by combining nodes in various ways, for example by organizing the clients into groups [13].

A broad classification of current trends in FL topology-related research can be found in [3], which classifies FL topology types into centralized (referred to also as star) [2], tree [14], hybrid [15], gossip [16], grid [17], mesh [3], clique [12] and ring [18]. Classic Federated Averaging [2], can be counted as an example of the centralized topology, involving only a single server independently communicating with each FL-participating client.

The TornadoAggregate algorithm, described in [15], combines star and ring topologies to form STAR-rings and RING-stars. One involves a central server performing periodic federated averaging combined with ring-based groups, while the other consists of a ring with star-based groups. Surprisingly, the first approach is significantly more successful, outperforming the RING-stars with regard to performance, while maintaining the same scalability as described in the aforementioned paper. This process does not require the setup of additional devices and so seems suitable for later reuse.

Many of the above-mentioned topologies use client grouping to manage the problems with heterogenous data. Moreover, this kind of mitigation method can also be used in combination with a centralized topology, in the form of centralized training with dynamic clustering implemented as IFCA in [19]. IFCA involves the simultaneous training of a given number of clusters, allowing for the dynamic creation of client clusters and models personalized for that cluster. However, some of the reported results were subpar due to the necessity of beginning the training with a warm start and accurate knowledge on the number of clusters present in the dataset [20]. In order to limit the occurrence of these issues, an improved version was developed. The new algorithm, SR-FCA, periodically reclusters the clients in a manner that leads it to be both more robust and less resource-intensive for edge clients.

In summary, there are many approaches to FL topology which result in different benefits and drawbacks. Some solutions focus on providing additional robustness to the system at the expense of decreased privacy and a more cumbersome setup. Others accept a communicational and computational overhead in exchange for the ability to use FL without selecting a single centralized server.

D. Scalability in FL

In our experiments, we have concentrated on the systems that are potentially easy to set up, scalable and able to withstand perturbations present in edge environments. Here,

a scalable FL system should be able to maintain high and effective performance in an massively distributed environment, that is, one with a very large number of clients [21]. Many topologies achieve scalability through the creation of local groups, which minimize the necessary frequency of the global aggregation rounds and, by extension, the communicational strain on the server [15].

III. Experimental Setup

In order to determine the best topology (both in terms of achieving the best possible performance and maintaining robustness to issues such as heterogenous data or client dropout) for the Assist-IoT pilots, our tests have been conducted on four potential solutions representative of the general trends. Those solutions are described visually in Figure 1. Their short summations can be found below.

1) Centralized: The centralized topology closely adheres to the original process of FL training from [2]. The server sends the model parameters to the clients, where they are trained for multiple iterations and subsequently aggregated on the server. We have decided to include this topology as a baseline for comparison with other, more sophisticated methods.

2) Centralized with dynamic clusters: The structure involves a server communicating with multiple clients. The main difference between this approach and a classic, centralized topology lies in the existence of multiple models (each of which is meant to be suitable just for a subset of the clients). After a given amount of training rounds, the server reclusters the clients based on the similarity of their weights (here estimated using Euclidean distance). The models are then aggregated separately for a given cluster [20]. Importantly, this implementation of SR-FCA (which stands for Successive Refine Federated Clustering Algorithm) aggregates the models using TrimmedMeanGD [22] instead of FedAvg, which provides additional robustness to the training process by removing outlier weights before aggregation. This method does not necessitate any previous knowledge about the number of clusters in the dataset nor additional computation on the client. The dynamic nature of its clustering algorithm causes this method to easily adapt to changes in client number and distribution. Unfortunately, SR-FCA does not necessarily increase the scalability of the solution in its current form.

3) Hierarchical: A hierarchical topology, known also as a tree topology [23], introduces a third type of node, apart from the client and server node to a centralized system: an intermediate (edge) node. In this case, the FL process begins with the server sending the model parameters to the edge nodes, which in turn send them to their clients. The clients train the model for a single iteration and send the results to the edge node, which aggregates those intermediate results. After this process repeats a set number of times, all the aggregated parameters from all of the edge nodes are once again aggregated on the main server [24]. As for the grouping of the clients to a given edge node, this simulation follows the heuristic introduced in [14] by spreading out groups of clients with similar data distribution between various clients.

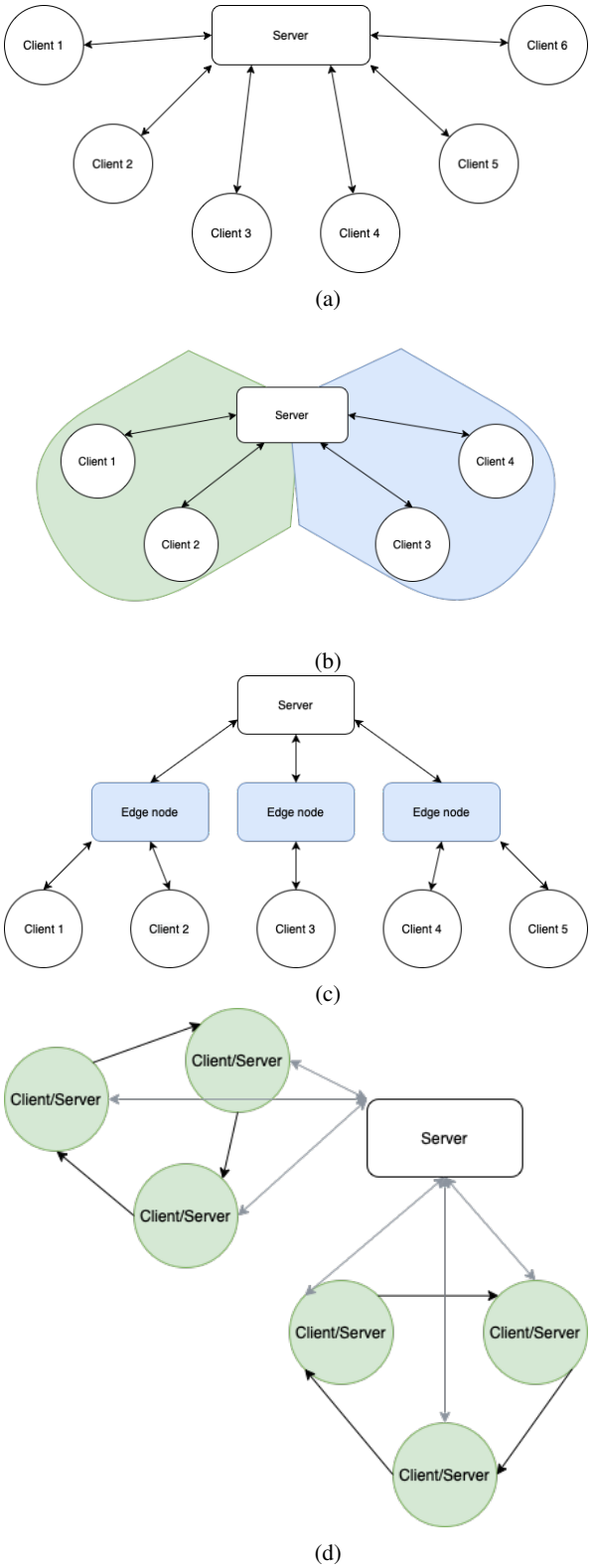


Fig. 1: A visualization of the FL system topologies investigated in this work

We have decided to test it as a more sophisticated and scalable FL topology, that nevertheless does not need computationally-intensive client clustering and does not introduce additional aggregation algorithms to Federated Averaging.

4) Hybrid: Tornadoes, or STAR-rings, is an especially promising approach presented in [15]. It combines a centralized solution with the existence of local, ring-based client groups. After the global server supplies the clients with starting parameters, the clients train the model and pass it on to the next client in their ring. In the next iteration, they accept an appropriate model from the previous client in the chain, train it for a given iteration on their own data and pass it on to the next client. After a set amount of inter-node iterations, all the model parameters from all of the clients are aggregated by the centralized server. Interestingly, since ring-based groups can be very susceptible to catastrophic forgetting in groups with high variance [15], a specialized clustering algorithm has been proposed by the authors. This algorithm requires access to client data distributions to compute the most optimal arrangement of ring groups, which may not be suitable for more private use cases. Additionally, it tends to be quite resource-intensive and frequently returns rings with significantly unequal numbers of nodes, which in some cases may complicate system maintenance. Although this FL topology requires using both an exhaustive client grouping algorithm and a more elaborate communication schema, the reported scalability of this method in environments with a large number of nodes is promising enough that we have decided to examine it further in our research.

A. Experiment Design

The experiments were conducted using the German Traffic Sign Recognition Benchmark Dataset [25], developed for a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks in 2011. The dataset incorporates 43 distinct classes divided into batches of size 16. The data was thoroughly shuffled and divided equally between the clients. Later, 80% of that dataset was used as the training data and 20% as the testing data. Independently of the local datasets, a global test set was placed on the server containing 12630 out of all the examples, with the remaining 39209 being divided between all the clients. In order to minimize the computations necessary for the simulation, the dataset has been rescaled to the size of 32 by 32 pixels.

The model used for the experiments consisted of 2 convolutional layers and a single dense layer. It was initially trained using the Adam optimizer with categorical cross-entropy loss without any gradient clipping. After the analysis of first experiments, gradient clipping was introduced for weights exceeding the value of 1.0. The clients were trained for 25 global rounds with 20 local iterations (the exact manner of conducting local iterations differed from topology to topology).

1) Client Grouping and Communication Schema: For the centralized training, no grouping of the clients was involved. Instead, the clients locally trained the model for one epoch on their own data and then sent those models to the server for aggregation, which constituted a full round. 25 of such training rounds have been conducted, with metrics such as aggregated loss, aggregated accuracy, global test set loss, and global test set accuracy being gathered after each of those rounds.

In the case of the centralized topology with dynamic clusters, the threshold λ of 5, size parameter t of 3 and β of 0.1 have been used. The Euclidean distance served as a metric to compute the differences between local weights. The clients have trained for 20 local iterations before each global round, and every 4 global rounds the clients were reclustered.

For the training of hierarchical FL a total of 5 intermediate nodes have been simulated, each managing 20 clients assigned to it. To sum up, the training was conducted on a 100 FL clients. Hierarchical FL involves a more intense communication protocol in the relation between the clients and the intermediate nodes: FedSGD [14]. For this reason, while the global communication schema between the edge nodes and the server has been maintained, the communication between the clients and edge nodes was much more frequent. Although this schema does increase the intensity of communication between nodes, it does not additionally overwhelm the server and, instead, maintains constant contact with edge nodes, which are presupposed to be much closer geographically located to the clients than the server.

Finally, for the hybrid topology, the number of 33 clusters was determined to be the most appropriate. This decision was influenced mainly by the suggestion placed in the original paper, highlighting the importance of small rings [15]. The original paper was also the source of the algorithm used for grouping the clients into clusters. The lengths of the resulting clusters vary from 1 to 8 clients per cluster. Additionally, the decision to conduct the experiments using a larger number of clusters did not influence the computational intensity of the process for the clients. It also did not add any overhead to the necessary communications between the clients and the server. Similarly to the hierarchical FL, the schema used here maintained a set number of global rounds with a set number of local rounds of training in between, here involving the clients accepting a new model, training it for one batch, and then passing it down the chain.

IV. Results and the diagnostic process

First tests conducted on IID data can be seen in Figure 2. Each one was conducted three times and averaged in order to obtain a smoother, more informative curve. Although two topologies, centralized (yellow) and centralized with dynamic clusters (blue), seem to be converging smoothly, there are suspicious perturbations that can be spotted both in the case of the hybrid topology (green) and hierarchical topology (purple). A potential explanation of the similar results of the centralized topology and centralized with dynamic clustering may stem from the fact, that in highly IID environments centralized topologies with dynamic clustering form just a single cluster and therefore are reduced to a simple centralized topology. Further examination reveals that each drop in the aggregated

accuracy for the hierarchical topology happened in a different run.

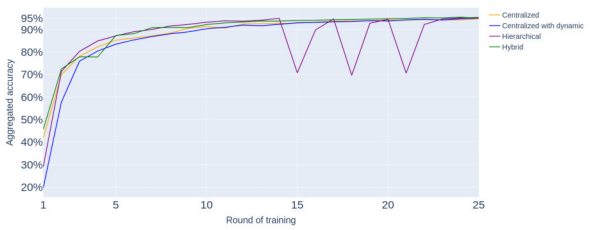


Fig. 2: The accuracy training curves for initial experiments

The issue has been further investigated in Figure 3a. The figure shows the mean aggregated loss measured for the clients belonging to a given cluster (differentiated with a color) after each local iteration and shown on a logarithmic scale. A cluster in our implementation of hierarchical FL includes all the clients performing their local aggregation on the same edge node, which means that all of the clients involved in the simulation are divided into 5 clusters with 20 clients each. The rise in aggregated test loss starts in iteration 201 with a global aggregation round (marked as a pink cross on the figure), and begins to drop after 221, so after another global aggregation round. It can be then observed that the sudden rise in aggregated was correlated with the adherence to a given cluster. Perhaps the frequent local aggregation rounds in hierarchical FL minimize the effectiveness of the Adam optimizer and cause gradient explosions to spread more effectively.

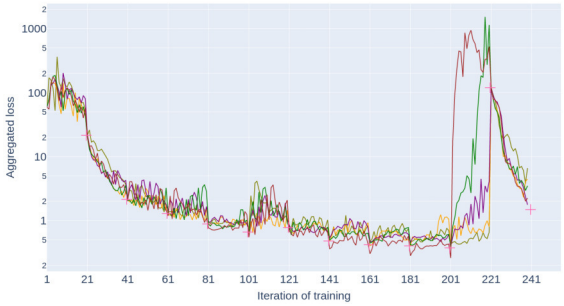
Figure 3b shows a makeshift metric, measuring the sum of the differences between obtained and trained weights for each client after each local iteration. The colors differentiating the adherence to a given node are maintained. The sudden increases in the weight differences correlate with the rise in aggregated loss both in the cluster affiliation and the iteration. These results confirm the existence of a gradient explosion problem.

After diagnosing the issue, an additional precaution of gradient clipping was applied to the model. New trials were then conducted to see if the learning curve improved. Figure 4a shows improvement in the form of a significantly smoother training process. Analogously, the difference in weights as shown on Figure 4b has stabilized and decreased significantly.

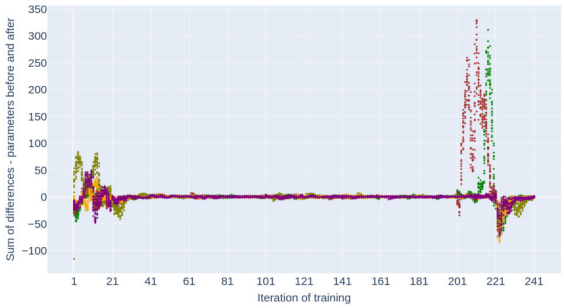
Repeating the first trial yields on Figure 5 slightly better, significantly smoother performance for all of the already mentioned FL topologies. An especially significant difference is visible for the hierarchical FL performance (purple), which suggests it to be an especially vulnerable topology to gradient explosions.

V. Discussions

Based on the usefulness of the additional metrics collected throughout the training in the diagnostic process, we propose continuous monitoring of the gradient scale of the local

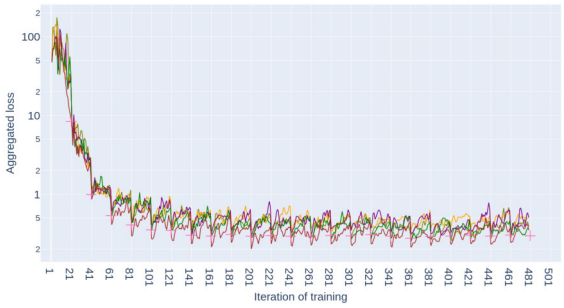


(a) Mean cluster aggregated loss for hierarchical FL

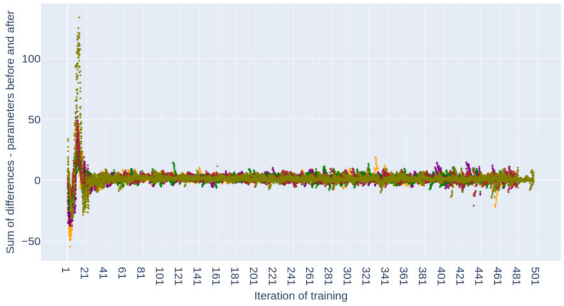


(b) Client weight differences for hierarchical FL

Fig. 3: Initial experiments



(a) Mean cluster aggregated loss for hierarchical FL



(b) Client weight differences for hierarchical FL

Fig. 4: Improved experiments (after the addition of gradient clipping)

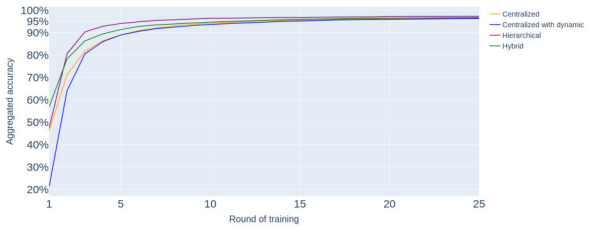


Fig. 5: The accuracy training curves for improved experiments

models through the regular computation of the gradient scale coefficient on the clients. The gradient scale coefficient is defined as follows.

$$GSC(k, l, f, \theta, x, y) = \frac{||J_k^l||_{qm}||f_k||_2}{||f_l||_2} \quad (1)$$

It measures the relative sensitivity of layer l with regards to random changes in layer k, measuring the size of the gradient flowing background relative to the size of the activations growing forward. A detailed explanation of this metric and how to use it to can be found in [6]. Its usefulness stems from robustness to network scaling, which introduces the possibility of result standardization.

We would like to measure the GSC of each of the client models after every iteration in order to use it to detect large, sudden shifts on the global level. These sudden shifts could then indicate the possibility of gradient explosion and prompt the developer to quickly recognize the problem without wasting needless resources for unsuccessful ML training.

VI. Conclusions

Although FL system and algorithm design remain popular research areas, the question on how to effectively enable the debug and maintenance of those systems is still largely unanswered. Our case study presents how a problem commonly encountered in classical ML may present in more complex FL topologies. We have also proposed a potential monitoring method for the early detection of such problems. All in all, we would like to stress the importance of the inclusion of such tools in distributed environments, where issues like client dropout or diverging distributions may be masking more fundamental problems.

References

[1] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *CoRR*, vol. abs/2009.13012, 2020. [Online]. Available: {https://arxiv.org/abs/2009.13012}

[2] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: {http://arxiv.org/abs/1602.05629}

[3] J. Wu, S. Drew, F. Dong, Z. Zhu, and J. Zhou, "Topology-aware federated learning in edge computing: A comprehensive survey," 2023. [Online]. Available: https://arxiv.org/abs/2302.02573

[4] "Pilot Scenario Implementation – First Version," 2022. [Online]. Available: {https://assist-iot.eu/wp-content/uploads/2022/05/D7.2_Pilot_Scenario_Implementation-First_Version.pdf}

[5] *Introducing Federated Learning into Internet of Things ecosystems – preliminary considerations*, 07 2022.

[6] G. Philipp, D. Song, and J. G. Carbonell, "The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions," 2018.

[7] A. Li, L. Zhang, J. Wang, F. Han, and X.-Y. Li, "Privacy-preserving efficient federated-learning model debugging," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2291–2303, 2022.

[8] Y. Liu, W. Wu, L. Flokas, J. Wang, and E. Wu, "Enabling sql-based training data debugging for federated learning," *CoRR*, vol. abs/2108.11884, 2021. [Online]. Available: https://arxiv.org/abs/2108.11884

[9] W. Gill, A. Anwar, and M. A. Gulzar, "Feddebug: Systematic debugging for federated learning applications," 2023.

[10] S. Duan, C. Liu, P. Han, X. Jin, X. Zhang, X. Xiang, H. Pan *et al.*, "Fed-dnn-debugger: Automatically debugging deep neural network models in federated learning," *Security and Communication Networks*, vol. 2023, 2023.

[11] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?" in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/13f9896df61279c928f19721878fac41-Paper.pdf

[12] A. Bellet, A. Kermarrec, and E. Lavoie, "D-cliques: Compensating noniidness in decentralized federated learning with topology," *CoRR*, vol. abs/2104.07365, 2021. [Online]. Available: {https://arxiv.org/abs/2104.07365}

[13] L. Chou, Z. Liu, Z. Wang, and A. Shrivastava, "Efficient and less centralized federated learning," *CoRR*, vol. abs/2106.06627, 2021. [Online]. Available: https://arxiv.org/abs/2106.06627

[14] N. Mhaisen, A. A. Abdellatif, A. Mohamed, A. Erbad, and M. Guizani, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 55–66, 2022.

[15] J. Lee, J. Oh, S. Lim, S. Yun, and J. Lee, "Tornadoaggregate: Accurate and scalable federated learning via the ring-based architecture," *CoRR*, vol. abs/2012.03214, 2020. [Online]. Available: {https://arxiv.org/abs/2012.03214}

[16] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distributed Applications and Interoperable Systems*, J. Pereira and L. Ricci, Eds. Cham: Springer International Publishing, 2019, pp. 74–90.

[17] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Federated learning for distributed spectrum sensing in nextg communication networks," 2022. [Online]. Available: https://arxiv.org/abs/2204.03027

[18] H. Eichner, T. Koren, H. B. McMahan, N. Srebro, and K. Talwar, "Semi-cyclic stochastic gradient descent," *CoRR*, vol. abs/1904.10120, 2019. [Online]. Available: {http://arxiv.org/abs/1904.10120}

[19] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," 2021.

[20] Harshvardhan, A. Ghosh, and A. Mazumdar, "An improved algorithm for clustered federated learning," 2022.

[21] M. Zhang, E. Wei, and R. Berry, "Faithful edge federated learning: Scalability and privacy," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3790–3804, 2021.

[22] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5650–5659. [Online]. Available: https://proceedings.mlr.press/v80/yin18a.html

[23] J. Wu, S. Drew, F. Dong, Z. Zhu, and J. Zhou, "Topology-aware federated learning in edge computing: A comprehensive survey," *arXiv preprint arXiv:2302.02573*, 2023.

[24] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Edge-assisted hierarchical federated learning with non-iid data," *CoRR*, vol. abs/1905.06641, 2019. [Online]. Available: http://arxiv.org/abs/1905.06641

[25] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012, selected Papers from IJCNN 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608012000457