VOLUME 18, N° 3 2024
Journal of Automation, Mobile Robotics and Intelligent Systems

Gradient Scale Monitoring For Federated Learning Systems

Submitted: 27th December 2023; accepted: 11th March 2024

Karolina Bogacka, Anastasiya Danilenka, Katarzyna Wasielewska-Michniewska

DOI: 10.14313/JAMRIS/3-2024/18

Abstract:
As the computational and communicational capabilities of edge and IoT devices grow, so do the opportunities for novel machine learning (ML) solutions. This leads to an increase in popularity of Federated Learning (FL), especially in cross-device settings. However, while there is a multitude of ongoing research works analyzing various aspects of the FL process, most of them do not focus on issues concerning operationalization and monitoring. For instance, there is a noticeable lack of research in the topic of effective problem diagnosis in FL systems. This work begins with a case study, in which we have intended to compare the performance of four selected approaches to the topology of FL systems. For this purpose, we have constructed and executed simulations of their training process in a controlled environment. We have analyzed the obtained results and encountered concerning periodic drops in the accuracy for some of the scenarios. We have performed a successful reexamination of the experiments, which led us to diagnose the problem as caused by exploding gradients. In view of those findings, we have formulated a potential new method for the continuous monitoring of the FL training process. The method would hinge on regular local computation of a handpicked metric: the gradient scale coefficient (GSC). We then extend our prior research to include a preliminary analysis of the effectiveness of GSC and average gradients per layer as potentially suitable for FL diagnostics metrics. In order to perform a more thorough examination of their usefulness in different FL scenarios, we simulate the occurrence of the exploding gradient problem, vanishing gradient problem and stable gradient serving as a baseline. We then evaluate the resulting visualizations based on their clarity and computational requirements. We introduce a gradient monitoring suite for the FL training process based on our results.

Keywords: *federated learning, exploding gradient problem, vanishing gradient problem, monitoring*

1. Introduction

Federated Learning (FL, [17, 25]) as a Distributed Machine Learning (DML) paradigm prioritizes maintaining the privacy of the devices (called clients). It aims to do so by leveraging the computing and communicational capabilities of the clients. A standard FL training process begins with the server initializing a machine learning (ML) model and subsequently communicating its weights to the clients.

The clients then use them to conduct local training and return their results to the server, where they are aggregated into a new global model. The whole process repeats multiple times until stopping criteria are met.

As of now, most of the ML models used for FL training are first designed in a centralized setting, with the developer having unrestricted access to a representative sample of the global dataset. Because of that, they are able to employ a variety of preexisting techniques and tools to make sure that the initial model architecture has been optimally selected. Many of the data preprocessing steps and hyperparameters developed in that initial phase form a base for later FL training. Unfortunately, this workflow can only be utilized for use cases where the representative global dataset can be constructed, excluding settings that demand additional privacy or just have largely distributed, heavily localized and client-specific data. In that case, the FL model development phase has to be conducted in a distributed environment over multiple runs, causing it to be potentially much slower. Distributed environments also involve the unexpected occurrences of other potential hazards in the form of sudden client dropout and differing client data distributions, causing the diagnosis of problems such as vanishing or exploding gradients to be significantly more difficult. This necessitates the development of effective tools of FL system diagnosis, for example through continuous monitoring of selected metrics. As this is a problem that affects the development and maintenance of FL systems, it can be understood as belonging to the domain of Federated Learning Operations (FLOps) [4], which aims to improve the FL lifecycle as a whole.

We have confronted the aforementioned issues during our work on the Assist-IoT project [1]. We have conducted trials to determine the most suitable FL topology to implement in the Assist-IoT project pilots centered around: (1) construction workers' health and safety assurance, (2) vehicle exterior condition inspection [6]. More specifically, it was important to provide a lightweight and scalable system for fall detection of construction workers in pilot 1 and automatic vehicle detection in pilot 2. In order to ascertain the best FL topology for the pilots, we have conducted a preliminary analysis of the issue [1] and selected 4 especially "promising" approaches in the form of the centralized, centralized with dynamic clusters, hierarchical, and hybrid architectures.

Our initial simulations have instead revealed some of those approaches (hierarchical and hybrid) to be especially sensitive to the exploding gradient problem, which in their cases presents itself as periodic drops in accuracy. The exploding gradient problem here is defined as a situation in which the gradient backpropagation in neural network training increases exponentially. This causes the training process to stall, with the resulting model deteriorating in some cases [31]. We have applied modifications to the experiment design in order to mitigate this problem. We have then described the whole process as a case study.

This article is an extension of the research presented in the conference paper [3]. We expand the theoretical part of this work, which now includes more information about the current state of FLOps with special importance given to the diagnostic tools. Descriptions of both the exploding and vanishing gradient problems are broadened as well, including both their common causes and mitigation techniques. A proposed gradient monitoring metric suite is designed by combining a modified version of the previously proposed Gradient Scale Coefficient (GSC) with the newly added average gradient per layer. The efficacy of the suite is tested in three simulated scenarios (vanishing gradient, exploding gradient, and baseline) for two selected topologies (centralized and hierarchical). Results are analyzed, both investigating the clarity of the visualizations produced by the suite as well as its necessary communication cost.

2. Related Works

2.1. Federated Learning Operations

Federated Learning Operations (abbreviated as FLOps) is a cross-discipline software development methodology. Its aims to improve the efficiency and quality of the development, deployment, and maintenance processes of FL systems [4]. As such, FLOps extends the principles devised for the purposes of MLOps and DevOps methodologies, such as continuous integration, deployment automation and model monitoring [18] to FL environments.

It is worth mentioning that the definition of FLOps formulated in [4] refers only to cross-silo environments. However, there are no clear reasons mentioned why it could not be extended to cross-device settings. On the contrary, there are many examples of cross-device business use cases such as the Gboard [37]. Although the particular activities composing the FLOps lifecycle in cross-device scenario may change, involving less negotiations between business entities and data interface formulations than in the cross-silo environments, the scenario still poses a significant challenge in terms of automation and operationalization. This work will focus on diagnosing problems caused by the gradient instability in cross-device FL systems. As effective solutions for FL diagnostics influence the efficiency and quality of FL development, it can therefore be considered as contributing to the research on FLOps.

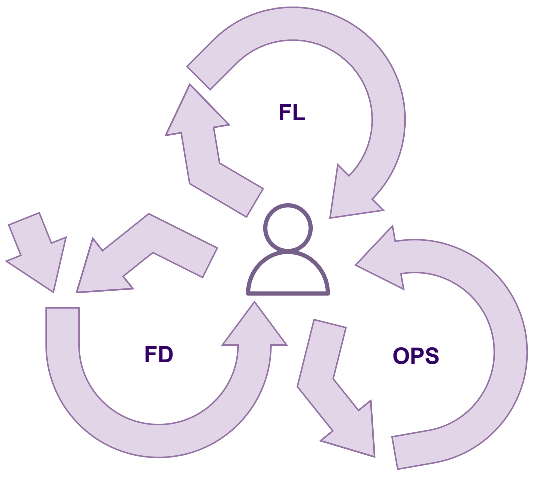

Figure 1. A simplified diagram of the FLOps process flows from [4]

The interaction between various FLOps flows is visualized in Figure 1. FD stands for Federated Design, which encompasses the processes of data analysis and model design. FL marks the flow of FL training, and OPS indicates the maintenance and monitoring of FL solutions deployed in production. Even though a given FL development process always begins with the FD phase, other phases can flexibly flow into each other based on the results achieved at a given stage. For example, a model that does not perform well may indicate the necessity of a return to FD, and insufficient performance metrics achieved during OPS may cause FL to restart.

Our research can be placed at the intersection of FD and FL, enabling an earlier transition from the latter to the former. It can be therefore understood as a means of optimizing the whole workflow in a holistic manner. Moving beyond the idea of optimizing a singular training process, effective FL diagnostic tools can shorten the length of the whole federated development process.

2.2. The State of FL Diagnostic Tools

As of now, the research on FL system diagnosis often centers around monitoring the clients in a secure and private manner in order to effectively distinguish those that are marked by their exceptionally bad performance [21] [24] [26]. As much as the solutions presented in the aforementioned works are interesting, they may not be sufficient to identify problems with a bad choice of hyperparameters or model architecture. FedDebug [11] offers the most comprehensive approach of all of those monitoring frameworks, enabling the developer to use metrics gathered throughout the training to replay previous rounds or set breakpoints. This is an effective solution to the problem of recognizing faulty clients.

However, some works involving other aspects of diagnosing FL systems can also be found. [20] provides a worthwhile contribution to the problem of FL model debugging by delineating how the integration of interpretable methods into FL systems may result in a potential solution, making it a very promising research direction. [7], on the other hand, concentrates on the software errors frequently encountered by the users of selected FL frameworks. Finally, Fed-DNN-Debugger [8] aspires to mitigate some of the problems affecting FL models (biased data, noisy data, or insufficient training) by influencing their local computation. Structure bugs, such as insufficient training as well as biased or noisy data, are beyond the scope of this solution. Fed-DNN-Debugger contains two modules, with the first one providing non-intrusive metadata capture (NIMC) and generating data that is then used for automated neural network model debugging (ANNMD).

2.3. The Exploding Gradient Problem

The problem of exploding gradient is caused by a situation, in which the instability of gradient values backpropagated through a neural network causes them to grow exponentially, an effect that has an especially significant influence on the innermost layers [31]. This problem tends to get occur more often the more depth a given ML architecture has, forming an obstacle in the construction of larger networks. Additionally, exploding gradient problem may in some cases be caused by the wrong weight values, which tend to benefit from normalized initialization [12]. When talking about activation functions, the problem may be avoided by using a modified Leaky ReLU function instead of the classic ReLU function. The reason for this behaviour lies in the addition of a leaky parameter, which causes the gradient to be non-zero even when the unit is not active due to saturation [27]. Another approach to stabilizing neural network gradients (for the exploding as well as the vanishing gradient problem) involves gradient clipping. The original algorithm behind gradient clipping simply causes the gradient to be rescaled whenever they exceed a set threshold, which is both very effective and computationally efficient [29].

There are other existing techniques, such as weight scaling or batch normalization, which minimize the emergence of this problem. Unfortunately, they are not sufficient in all cases [31]. Some architectures, for instance fully connected ReLU networks, are resistant to the exploding gradient problem by design [14]. Nonetheless, as these architectures are not suitable for all ML problems, this method is not a universal solution.

2.4. The Vanishing Gradient Problem

The reverse of the exploding gradient problem, the vanishing gradient problem is considered one of the most important issues influencing the training time on multilayer neural networks using the backpropagation algorithm. It appears when the the majority of the constituents of the gradient of the loss function approaches zero. In particular, this problem mostly involves gradient layers that are the closest to the input, which causes the parameters of these layers to not change as significantly as they should and the learning process to stall.

The increasing depth of the neural network and the use of activation functions such as sigmoid makes the occurrence of the vanishing gradient problem more likely [32]. Along with the sigmoid activation function, the hyperbolic tangent is more susceptible to the problem than rectified activation functions (ReLU), which largely solves the vanishing gradient problem [13]. Finally, similarly to the exploding gradient problem, the emergence of the vanishing gradient problem has been linked to weight initialization, with improvements gained from adding the appropriate normalization [12].

2.5. Advances in Research on Topology of Federated Learning

The default, centralized network topology used for a FL system, which involves a single powerful cloud server communicating with a federation of clients located on edge and IoT devices may not be the most suitable solution for all use cases [35]. Some require efficient communication, which may be more effectively provided by the solutions that have either reduced the importance of the server or removed it all together [2]. Others focus on leveraging network topology to minimize problems cause by data heterogeneity. There also those that attempt to combine the two approaches described above by carefully grouping the clients [5].

[35] includes a catalogue of many commonly encountered trends in research involving FL topology, classifying FL topology types to 7 categories, including centralized [25], tree [28], hybrid [19], gossip [16], grid [33], mesh [35], clique [2], and ring [9]. Here, Federated Averaging described in [25] is an example of the centralized topology. TornadoAggregate, on the other hand, is understood as belonging to the hybrid category due to it constructing STAR-rings and RING-stars by combining star and ring topologies [19]. STAR-rings indicates the existence of a server, which performs regular client weight aggregation along with ring-based groups. RING-stars constructs a large global ring, with small centralized groups conducting local computation and passing it periodically to other groups in the chain. Out of those two, STAR-rings receives much better performance results while maintaining the same scalability.

Some systems combine different topological approaches in order to create a more responsive system, that can, for instance, adaptively respond to problems with heterogenous data. IFCA [10] integrates a centralized topology with dynamic clustering by periodically grouping the clients and simultaneously training a personalized model for each of the obtained groups. Unfortunately, as this method necessitates a warm start to the training and prior knowledge about the number of clusters necessary, it leaves significant space for improvement [15]. This improvement comes in the shape of SR-FCA, which can automatically determine the right amount of clusters, making it more robust and resource-efficient.

All in all, there is a wide variety of FL topological approaches developed to prioritize different aspects of the system, such as scalability, robustness, or privacy. These deviations in priorities make the comparison of those approach especially difficult, as they often involve modifications not only to the architecture, but to the clustering or aggregation algorithms as well. Moreover, it should be noted that many of the works involving the topic of FL topology focus on a limited range of experiments aiming to achieve the best performance. As a result, further issues such as the expression characteristics of the exploding gradient problem in selected topologies and how it differs from ML in most cases remain unexplored.

3. Experimental Setup

Our initial goal for the case study was to ascertain the best topology for the Assist-IoT pilots according to our criteria of maintaining the best possible performance while exposed to negative factors such as client dropout or non-IID client data distribution. We have also taken into account the ease of infrastructure setup for a given topology and the scalability of the whole system. Here, scalability in FL systems is understood as the capability to maintain stringent performance requirements in environments that are massively distributed [39], that is, contain a very large number of clients.

To achieve this goal we have selected four promising solutions, each representing a different approach to the problem and, therefore, allowing us to examine a broad range of trends. The topologies of those solutions are visualized in Figures 2, 3, 4, and 5. Their accompanying descriptions can be found in sections 3.1, 3.2, 3.3, and 3.4, respectively.

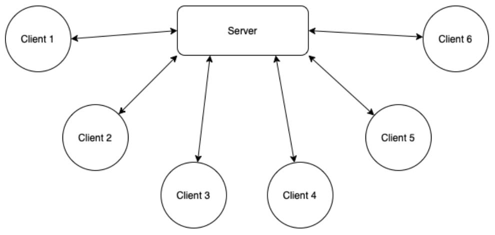

Figure 2. A visualization of the FL centralized topology

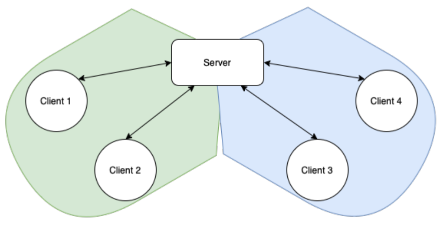

Figure 3. A visualization of the FL centralized topology with dynamic clusters

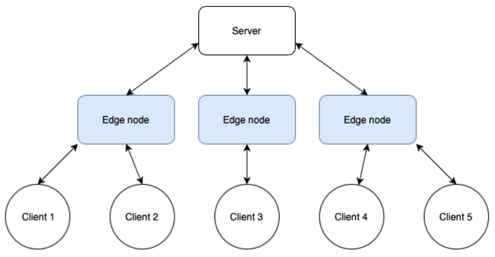

Figure 4. A visualization of the FL hierarchical topology

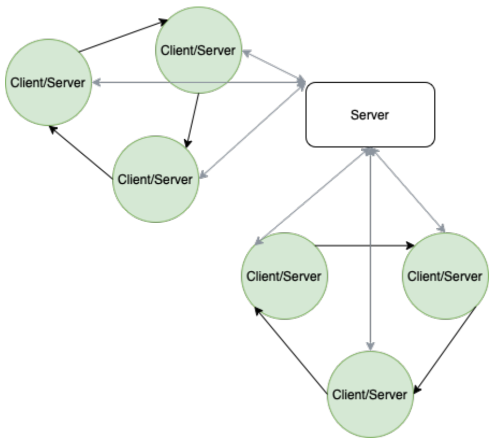

Figure 5. A visualization of the FL hybrid topology

3.1. Centralized

The centralized topology has been introduced along with the concept of FL as a paradigm in [25]. This topology consists of a server (which sends the initial model parameters to the clients and periodically aggregates their results to construct a new global model) and multiple clients (which handle local computation). As such, it is often distinguished by its asymmetric data flow and information concentration on the server, which may result in potential risks and unfairness [22]. As the centralized topology is often considered a default in FL systems, we have included it as a baseline for comparison with newer, potentially more scalable solutions.

3.2. Centralized with Dynamic Clusters

On the surface, the centralized topology with dynamic clusters strongly resembles the centralized topology as described in section 3.1. In involves periodic communication of a singular server with a group of clients, where the clients handle model training and the server manages weight aggregation. However, this topology additionally includes a dynamic component in the form of multiple personalized models (each of the models is developed only for a fraction of the clients).

Moreover, the assignment of the clients to a particular model varies as well. It is recomputed regularly by the server to ensure that it is still optimal. The assignment is based on weight similarity, which is estimated using the Euclidean distance. The weights of each model are then computed only based on the clients assigned to a given cluster at the given moment [15]. This version of the centralized topology with dynamic clusters includes an additional improvement. As it is implemented according to the SR-FCA (Successive Refine Federated Clustering Algorithm), it uses the Trimmed-mean-based Gradient Descent [38] algorithm instead of Federated Averaging for weight aggregation. As Trimmed-mean-based Gradient Descent excludes outliers from aggregation, its employment causes the system to be more resistant to abnormal client behaviour such as Byzantine failure [38]. SR-FCA can respond to environmental changes and adjust to varying client data distributions without any prior information about the necessary number of clusters nor additional local computation. It also provides flexible personalization, automatically producing multiple models for differing groups of clients. However, it is important to mention that it may not lead to an increase in scalability.

3.3. Hierarchical

Hierarchical topology (described as tree topology in [36]) innovates the previously described centralized topology by adding a third category of device: the edge node. The edge nodes act as an intermediary server between the main server and clients, aggregating local model weights from all clients assigned to them after each iteration and subsequently passing the aggregated models onto the server each global round. The server then aggregates the edge results and forms a new model, that is later communicated to the clients only for the process to begin again [23]. In order to maintain convergence, the overall data distribution of the clients allotted to each edge node should resemble the global distribution as much as possible. This assumption is maintained in our simulation using the method from [28], which advises to divide groups of clients with similar distributions between multiple edge nodes. We have been motivated to select the hierarchical topology for our trials by its combination of simplicity and scalability, as it manages to significantly reduce the communicational load put on the server while avoiding computationally-intensive clustering algorithms.

3.4. Hybrid

For our final example of a hybrid topology we have decided to examine Tornadoes (described also as STAR-rings in [19]). Tornadoes integrates a centralized architecture with local computation performed inside particularly formed ring-based groups. After each global round the clients receive a new model from the server. They train it for an iteration and pass the results to the next client in their ring. In turn, they receive a new model from the previous client in their ring, which they train and pass onto the next client. This process repeats of a set number of local iterations. Afterwards, the server aggregates all local models from all clients belonging to all rings, forming the new model which is later communicated to the clients, letting the process restart. This hybrid topology provides additional scalability to the system by decreasing communication between the server and clients without increasing the necessary infrastructure to do so.

Moreover, the decision to use decentralized local groups instead of edge servers makes the system as a whole more robust to failure. However, as ring-based groups with high variance between client distributions are vulnerable to catastrophic forgetting [19], the clients have to be divided into groups using dedicated algorithms. The information required to divide the clients differs between approaches, with some being significantly less private than others. The aforementioned grouping algorithms may in some cases be very compute-intensive as well. All in all, in spite of potential drawbacks visible in the design of Tornadoes, we have decided to investigate potential scalability increase it might provide.

4. Initial Experiment Design

We have elected to use the German Traffic Sign Recognition Benchmark Dataset [34] for our simulations, as it is both lightweight and less frequently used than the datasets mentioned in [25], [15], [28], and [19] and would therefore provide a complementary source of information to the research presented in those works. The dataset has been designed for a multi-class, single-image classification challenge organised as a part of the International Joint Conference on Neural Networks in 2011. It consists of 43 distinct classes and contains a global test set with 12630 examples and training set with 39209 examples. For the purpose of our experiments, the training set has been shuffled and divided equally between 100 clients, with 80% of each client's local data being used for training and 20% for testing. The global test set was used to compute model accuracy each round, whereas the local test sets were employed to calculate aggregated accuracy. As a way to reduce the computational cost of the training process, the dataset as a whole has been resized to 32 by 32 pixels.

The neural network architecture prepared for the simulations contained 2 convolutional layers and 1 dense layer. At first, it utilized the Adam optimizer without any gradient clipping, which after modifications changed to gradient clipping for weights exceeding the value of 1. In both cases, the hyperparameters used included an initial learning rate of 0.001, β_1 of 0.9, β_2 of 0.999, ϵ of 10^{-7}, batch size of 16, 25 global rounds, 20 local iterations, and categorical cross-entropy used as a loss function.

4.1. Client Grouping and Communication Schema

The clients were not grouped at all for the experiments investigating the centralized topology. The experiments have been conducted for 25 full rounds, each including every client training for 20 iterations on local data before sending the weights to the server. The clients then received the new weights in order to compute on them metrics such as accuracy and loss, which were then aggregated on the server to obtain aggregated accuracy and aggregated loss, respectively. The server also calculated global test set accuracy and global test set loss.

The parameters used for clustering in simulations conducted on the centralized topology with dynamic clusters included size parameter t of 3, β of 0.1 have been used, and threshold λ of 5. Here, local training on the clients has similarly lasted 20 local iterations, with client reclustering being performed after each 4 global rounds.

For the purposes of hierarchical topology simulation 5 edge nodes have been used with 20 clients assigned to each. As the inclusion of edge nodes minimizes the additional communicational load on the server, a more intense communication protocol has been used for local computation. To accurately recreate the approach presented in [28], after each of the 20 local iterations every client communicated its resulting weights to its assigned edge nodes, where they were aggregated and sent back to the clients. This process has been repeated for 25 global rounds.

In order to accurately reproduce the selected hybrid topology while minimizing the computational intensity of the simulations, all of the clients were grouped into 33 rings of varying length using the algorithm described in [19]. Each global round has contained 20 local iterations, with all of the clients accepting the weights sent by the previous client in the ring, training them for one iteration, and passing them onto the next client in the ring. Afterwards, the global model is formed by aggregating all of the local models on the server.

5. Initial Results and the Diagnostic Process

Figure 6 shows the initial test results. Two of the selected topologies, centralized (yellow) and centralized with dynamic clusters (blue), converged to a satisfying solution with minor disturbances, which cannot be said about about the hybrid topology (green) and hierarchical topology (purple). As each of the experiments was repeated three times to form a more robust, smoother curve, the periodic drops in aggregated accuracy visible for the hierarchical topology cannot be explained by the existence of an outlier. Instead, each of the drops occurs for a different run. Additionally, the resemblance in results obtained for the centralized topology to those for the centralized with dynamic clusters may stem from the IID client data distribution in the simulated environment. In these conditions the centralized solution with dynamic clusters tends to form a single client group, behaving similarly to a plain centralized topology.

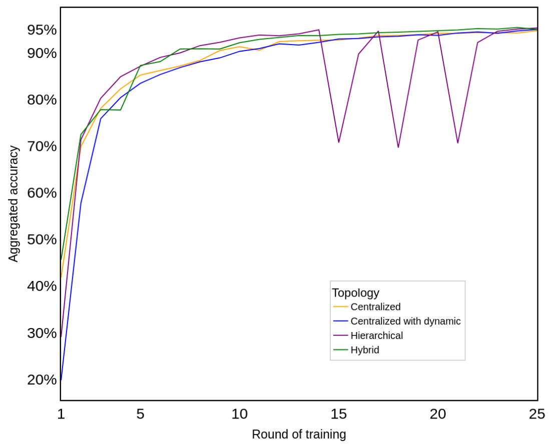

Figure 6. The accuracy training curves for initial experiments

Figure 7 depicts our further inquiry into the issue. It visualizes mean aggregated loss for each cluster of clients measured after each local iteration for a hierarchical topology simulation, where a cluster means all of the clients assigned to a given edge node. In order to clearly distinguish between the clusters they were assigned different colors. A sudden increase in mean aggregated loss can be observed after iteration 201 for one of the clusters, with some of the other clusters experiencing similar instabilities. They may have been caused by the combination of an Adam optimizer, which was designed with the assumption of a centralized ML environment, with frequent local communication between the client and the edge node, which causes gradient instabilities. Interestingly, the next global aggregation in iteration 221 seemingly partially mitigates the issue.

In order to verify that the drops in accuracy for the hierarchical topology were caused by gradient instabilities, we have designed a makeshift metric. The metric subtracts weights after and before local training in each iteration for every client, and then sums up all of those differences. Figure 8 visualizes the results, with the assignment of a color to a given cluster exactly the same as in Figure 7. The sudden rise in the values of the metric for each of the clusters correlated with the increasing aggregated loss, supporting the hypothesis about it being a gradient explosion problem.

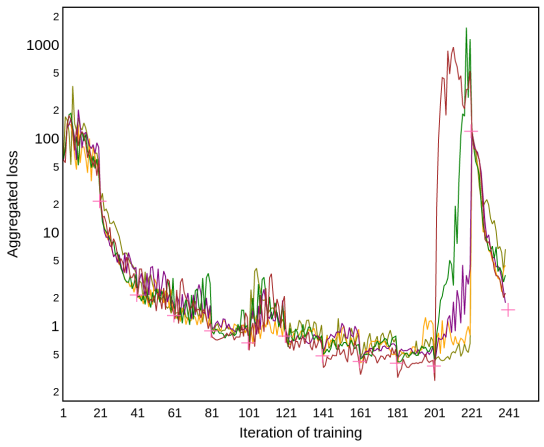

Figure 7. Mean cluster aggregated loss for hierarchical FL in initial experiments

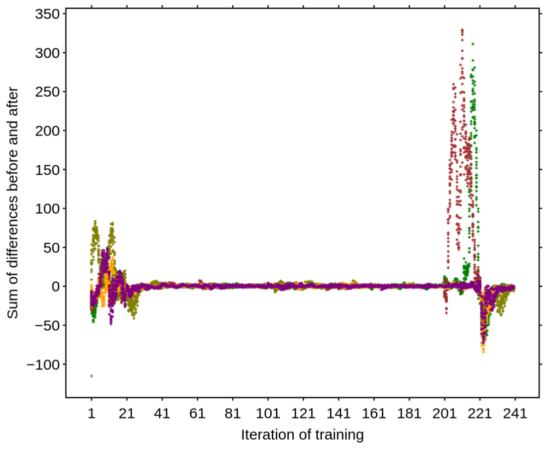

Figure 8. Client weight differences for hierarchical FL in initial experiments

Having formulated an initial explanation, the training process has been modified to include gradient clipping as a sample method for stabilizing the training. Subsequently, the trial has been repeated to ensure that our explanation has been sufficient. Figures 10 and 9 both show findings agreeing with this statement in the form of smaller and more stable values in the case of Figure 10 and no sudden, extreme loss increases in the case of Figure 9.

Figure 11 recreates the first trial with the modified training process, achieving a much smoother curve for all of the topologies. The hierarchical topology (purple) experiences the most visible improvement, which indicates it to be potentially the most vulnerable to the problem of exploding gradient.

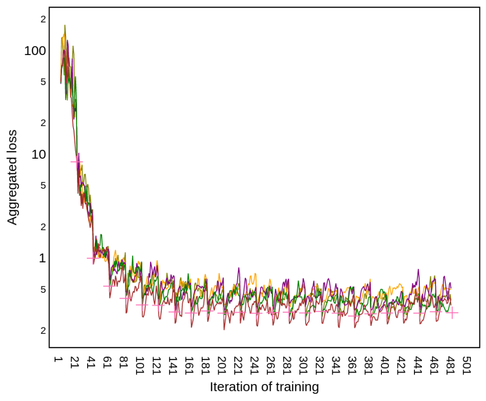

Figure 9. Mean cluster aggregated loss for hierarchical FL in improved experiments

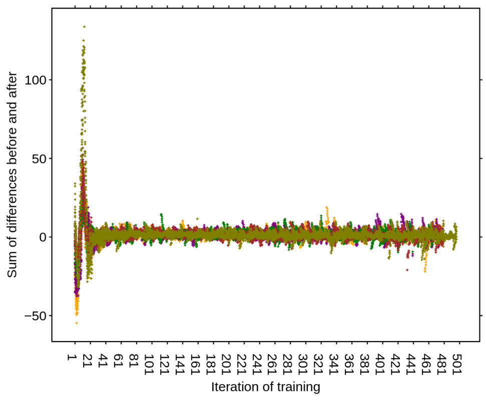

Figure 10. Client weight differences for hierarchical FL in improved experiments

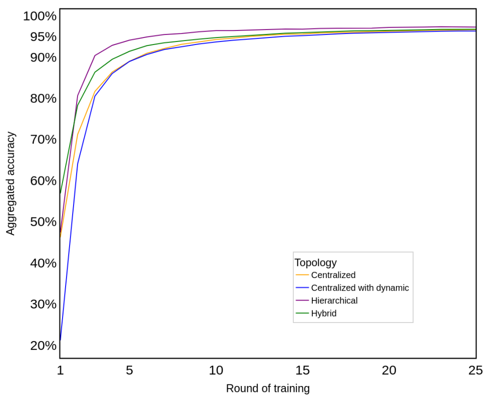

Figure 11. The accuracy training curves for improved experiments

6. Preliminary Metric Choice

6.1. Gradient Scale Coefficient

In order to maximize the usefulness of the additional metrics collected throughout the training in the diagnostic process, in the initial work we have proposed continuous monitoring of the gradient scale of the local models through the regular computation of the gradient scale coefficient on the clients. The gradient scale coefficient is defined as follows.

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

It measures the relative sensitivity of layer l with regards to random changes in layer k, capturing the size of the gradient flowing background relative to the size of the activation values growing forward.

A detailed explanation of this metric and how to use it to can be found in [31]. Its practicality stems from its robustness to network scaling, which introduces the possibility of result standardization (although the validity of this property needs to be tested further, as the original work focused only on limited neural network architectures). Additionally, the ability to summarize the degree in which the gradient is currently vanishing or exploding could contribute to an effective visualization for a potential future user comparing multiple FL runs on a single plot.

In our work, we use a version of GSC modified according to the equation shown below:

$$GSC(L+1,0) = \frac{\frac{1}{\sqrt{n}}||J_{L+1}^{0}||_F\,||f_{L+1}||_F}{||f_0||_F} \tag{2}$$

As the architecture used in our problem (convolutional neural network) differed from the architectures tested in [31] (multilayer perceptron), the original formula did not account for the difference in layer shapes. We perform necessary modifications on the GSC while keeping it as close to the original solution as possible through changing the type of the norms used from second order to Frobenius and omitting the bias. Inspired by the diagrams presented in [31], in order to extract as much information as possible while minimizing the computation, we decide to compute the GSC only for the interaction between the first and last layer.

6.2. Average Gradient Per Layer

To add a source of information about specific layers while maximizing the frugality of our resulting metric suite, we set out to include the average gradient of the weights per layer. We do not include bias in our computations. While simple, we suspect this information to be beneficial in cases when the extent to which a specific layer is affected by the vanishing or exploding gradient problem may play a role. For instance, when determining to which degree the model suffers from a vanishing gradient problem, it may be helpful to analyze whether the gradient values remain close to 0 only for a single layer, or multiple layers close to the input. In this approach, we are inspired by [12]. Whereas this work uses standard deviation intervals of weight gradients per layer in time to check whether their proposed normalization affects the gradient throughout training, we can focus on the general scale of the gradient expressed through the average value. Utilizing this approach instead of, for instance, gradient histograms per layer, will allow us to clearly depicting the time component while minimizing the communication load of the metric.

7. Extended Experimental Setup

7.1. Scenario Description

To simulate the three scenarios of exploding gradients, vanishing gradients, and that of the baseline, appropriate modifications are applied to the model architecture and training procedure according to the theory presented in sections 2.3 and 2.4.

The baseline scenario is analogous to the corrected model described in section 4. It consists of two convolutional layers and one dense layer. The activation function used for the two convolutional layers is Leaky ReLU. The weights of the convolutional layers are initialized by the Glorot Uniform initializer. In this example, the Adam optimizer is modified to include gradient clipping, with the threshold for each weight being that the norm of its gradient does not exceed 1.

The exploding gradient scenario is modified to include ReLU as activation function instead of Leaky ReLU. The weights of the convolutional layers are initialized using the uniform distribution with values ranging from 8 to 10. Additionally, the gradient clipping mechanism is removed. Aside from the aforementioned changes, the architecture of the model does not differ from the baseline.

The weights for the convolutional layers in the vanishing gradient scenario are initialized using the uniform distribution with values ranging from 4 to 7. Tanh, or hyperbolic tangent, is used as the activation function for the convolutional layers. The gradient clipping is removed in this scenario as well. Aside from that, there are no further changes applied to the model in this scenario compared to the baseline.

7.2. Metric Measurement and Other Modifications

The algorithms for computing the GSC and average gradient per layer used in this work are described more in depth in section 6. Here, we will focus on the details of integrating the algorithms into FL topologies. In both centralized and hierarchical topology, the GSC and average gradient per layer is computed after each local iteration using the whole training set of the client in order to fully capture changes in the gradient brought on by local computation. As the hierarchical architecture involves the existence of multiple local iterations per each global round, we gather metrics after each of these iterations. Then we analyze the diagrams constructed from all of the measurements, as well as only those computed for the last iteration before a global aggregation round. This action is performed in order to determine the extent of the information loss that could result from this much more communicationally and computationally effective scheme.

There is an open possibility of integrating the computation of GSC and average gradient per layer with the local training by using the gradients that are already computed as a part of the training. We decide to avoid it due to the assumptions presented in [31], which referred to the whole available dataset instead of just a batch. However, this opportunity may still be explored in scenarios with small local training sets available for each client.

To ensure that we minimize the impact of random factors on the results obtained through experiments, each of the trials has been conducted three times. The final result of a selected trial is a mean of all of the runs with additional information included about the differences between them. Similarly, to increase the quality of the results, each experiment has been conducted for 50 global rounds instead of 25. In order to minimize the amount of computation necessary, we have decided to focus on two of the four FL topologies described in this work - the centralized topology, as it's the most commonly used and can therefore serve as an effective baseline, and the hierarchical topology, as it is the most vulnerable to the exploding gradient problem from all of the initially investigated examples. Apart of the modifications described in this section, the extended experiments are conducted according to section 4.

8. Extended Results

The first part of our analysis focuses on examining the behaviour of our scenarios and ensuring that it agrees with our assumptions embedded in the experiment design. To accomplish this, we look at the training accuracy curves.

Figures 12, 13, and 14 depict the training process conducted for the baseline, exploding gradient, and vanishing gradient scenario, respectively. In all three examples, the FL topology employed was centralized. In the first, baseline scenario, the depicted curve is smooth and reaches an aggregated accuracy exceeding 95%. In the second scenario depicted in Figure 13, the final accuracy is significantly lower, with a jagged curve and much more pronounced differences between runs marked in the figure by the light blue color.

This indicates instability inherent in the exploding gradient scenario. Finally, the training process visualized in Figure 14 is fully dysfunctional, with extreme variations in aggregated accuracy, which is nevertheless unable to exceed the threshold of 6%. This marks a scenario with a vanishing gradient problem so severe that the model is functionally unable to learn.

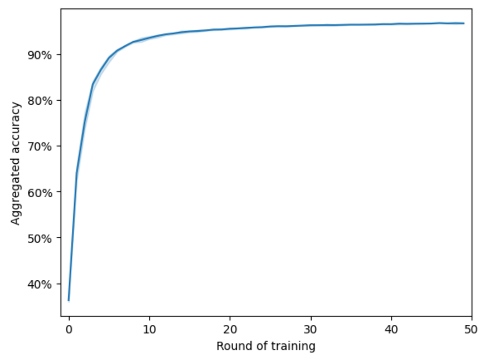

Figure 12. The accuracy training curve for centralized FL in the baseline scenario

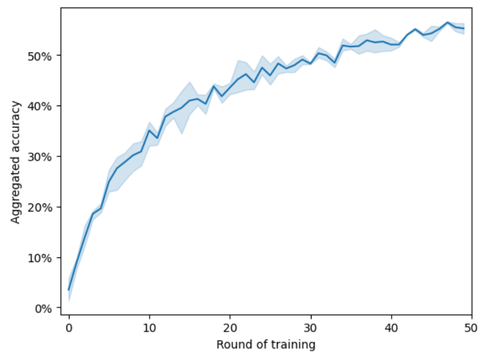

Figure 13. The accuracy training curve for centralized FL in the exploding gradient scenario

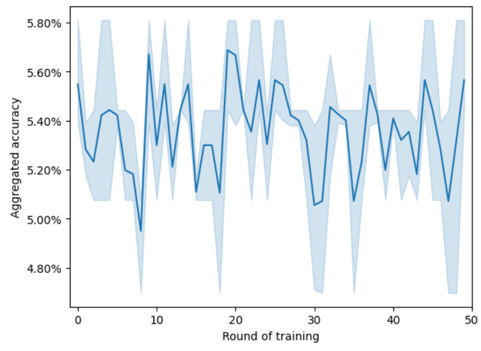

Figure 14. The accuracy training curve for centralized FL in the vanishing gradient scenario

Figures 15, 16, and 17 represent similar training processes for the hierarchical FL topology. Here, the training instabilities are even more pronounced for the exploding and vanishing gradient scenarios (Figures 16 and 17), resulting in much more extreme changes in accuracy between training rounds and runs.

Moving onto the analysis of our metric suite, in Figure 18 we can observe the GSC values for all three scenarios reenacted in a centralized FL architecture. Here, the vanishing gradient problem is marked by a GSC of 0 unchanging for the duration of the entire run. This marks is as easily distinguishable without any additional knowledge about the current training accuracy of the gradient stability in other scenarios.

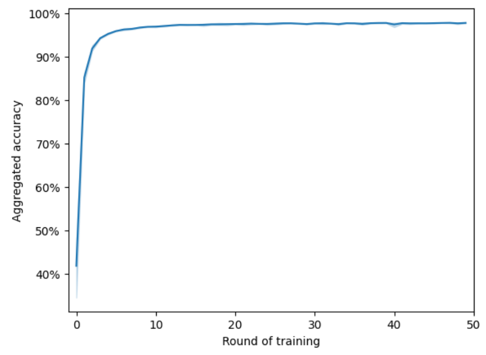

Figure 15. The accuracy training curve for hierarchical FL in the baseline scenario

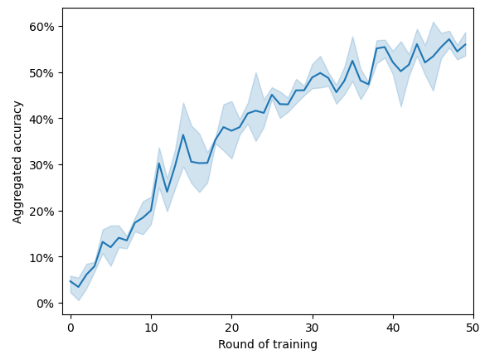

Figure 16. The accuracy training curve for hierarchical FL in the exploding gradient scenario

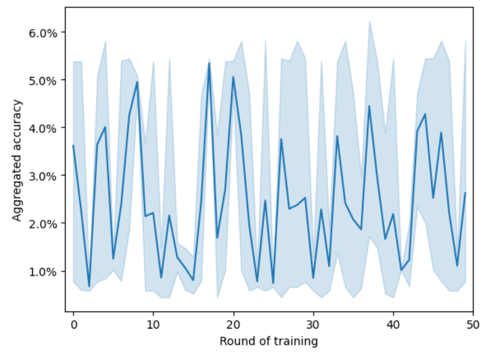

Figure 17. The accuracy training curve for hierarchical FL in the vanishing gradient scenario

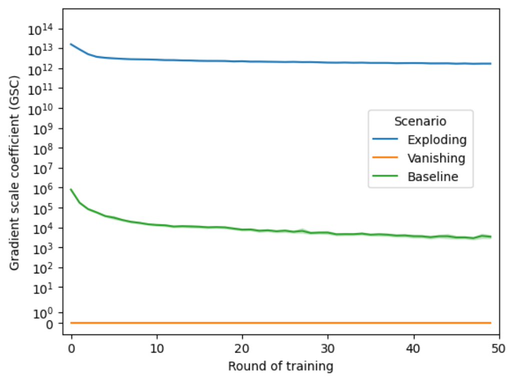

Figure 18. Gradient scale coefficient (GSC) values for centralized FL in different scenarios

Additionally, the GSC values allow all the scenarios to be visually identifiable from each other, with the scale of the values for the exploding gradient being visibly larger than for the baseline. This may be beneficial for the iterative process of conducting multiple FL training runs, as it would provide a developer with information about the current gradient scale in the context of other runs. For example, for a developer seeking to fix an exploding gradient problem and testing a potential solution it would be helpful to know the scale of the GSC when compared to previous runs.

Unfortunately, it is not obvious whether the GSC of a single run would be sufficient to diagnose it as suffering from a vanishing or exploding gradient problem. There is some discussion about it being potentially true in [31]. However, as the aforementioned work focuses on the multilayer perceptron architecture, further research involving other model architectures is still necessary.

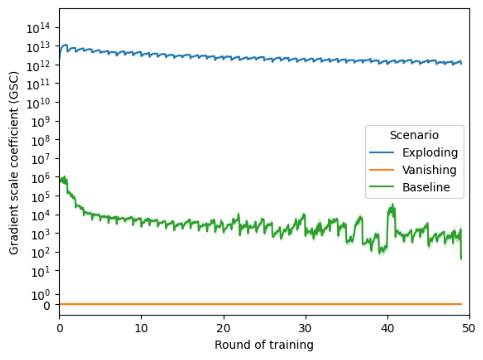

Figure 19. Gradient scale coefficient (GSC) values for hierarchical FL in different scenarios

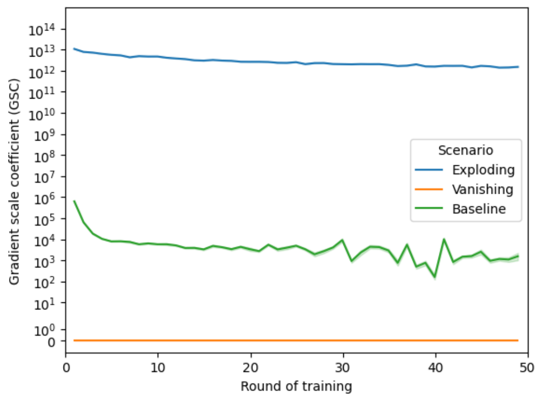

Figure 20. Gradient scale coefficient (GSC) values measured for the last local iteration for hierarchical FL in different scenarios

Figure 19 showcases GSC values for different test scenarios reenacted in a hierarchical FL system. The GSC in these experiments is similar to analogous results for the centralized topology both in scale and clear separability between scenarios. Additionally, the GSC depicted in Figure 19 seems to maintain traits specific to the hierarchical topology, such as periodic changes in gradient caused by global weight averaging and special vulnerability to gradient instabilities visible for the baseline scenario.

Figure 20 depicts a simplified version of the previous diagram containing only the GSC values measured after the last local iteration before the global computation round. It can be noted that Figure 20 effectively preserves most information from Figure 19, including the scale and stability of GSC for a given scenario, omitting only the visible indicators of periodicity.

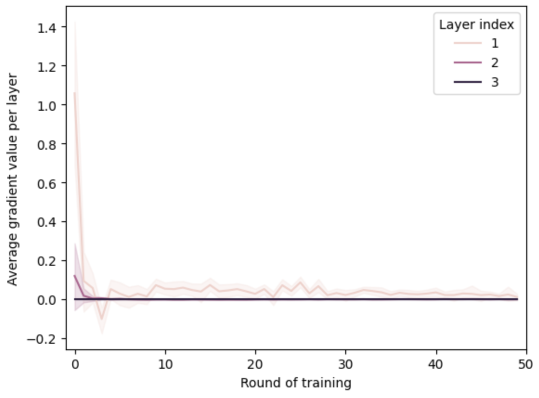

Figure 21. Average gradient value per layer for centralized FL in the baseline scenario

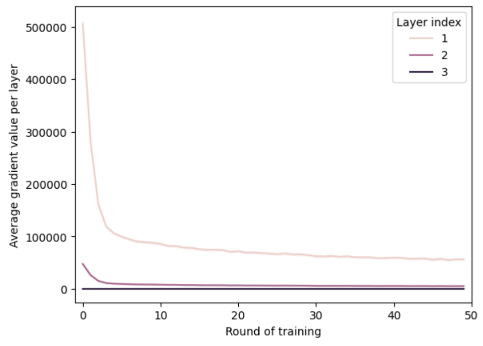

Figure 22. Average gradient value per layer for centralized FL in the exploding gradient scenario

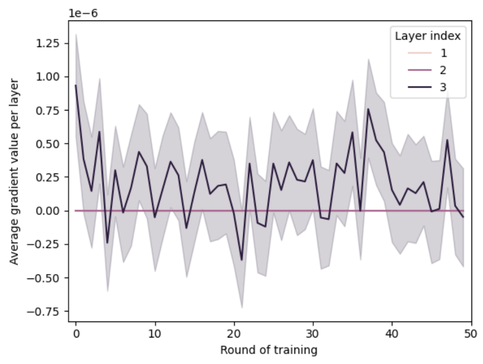

Figure 23. Average gradient value per layer for centralized FL in the vanishing gradient scenario

Figures 21, 22, and 23 present the average gradient value for a given layer for experiments conducted on a centralized FL system. The layers are numbered from the input to the output, marking 1 as the layer closest to the input and 3 as the layer closest to the output. This is a simple yet effective visualization, as it allows the viewer to easily compare gradient values between layers.

In Figure 21 (baseline scenario), this means that average gradient values per layer are both relatively low and close to each other. Although the average gradient value for layer 1 is often larger than for value 3, layer 1 frequently shifts and intersects with layer 3. We can contrast it with Figure 21 (exploding gradient scenario), where the average gradient of layer 1 is noticeably greater than layer 2 and 3 with a large difference in scale. Figure 22 (vanishing gradient scenario) marks a training process where layers 1 and 2 are extremely close to 0 throughout the whole training, with the values of layer 3 varying largely between iterations and runs. These clear differences in plots indicate, that average gradient values per layer as a metric may be enough to effectively recognize an exploding or vanishing gradient problem in FL systems.

Figures 24, 25, and 26 move onto depicting average gradient values per layer for the hierarchical topology. Here, the results are similar to the scenarios simulated for the centralized system, if notably less readable due to a larger amount of local iterations (and therefore also gradient measurements). Interestingly, periodic drops in the average gradient of the first layer depicted in Figure 25 seem to confirm our prior suspicion about global weight aggregation serving as a form of regularization.

Figures 27, 28, and 29 are very similar to Figures 24, 25, and 26, with the only difference being the amount of information depicted. Figures 27, 28, and 29 contain only the average gradient values measured per layer after the last local iteration in each round. Interestingly, these limitations seem to influence the visualizations positively. The plots are now easier to read, with Figure 27 depicting average gradients that are more clearly similar in scale. The only important information lost is the periodicity in Figure 28, which is not necessary to determine it to be an example of the exploding gradient as the average values of layer 1 remain visibly larger from layer 3.

The final gradient scale monitoring framework therefore includes the GSC visualization (as shown, for instance, in Figure 20) to enable easy and readable gradient scale comparison for multiple runs of the system, as well as the average gradient per layer, which provides simple gradient problem diagnosis for particular runs. The GSC should be displayed using the logarythmic scale. To minimize additional computational load caused by the monitoring, the metrics should be computed only for the last local iteration in centralized topologies with local groups (such as hierarchical or hybrid).

9. Conclusion

Even though there is plenty of research focused on many different aspects of the FL paradigm, we identify a potential gap in the topic of effective diagnosis and monitoring of FL systems. For instance, our initial experiments showcase how differently a problem commonly encountered in ML may present in more sophisticated FL topologies.

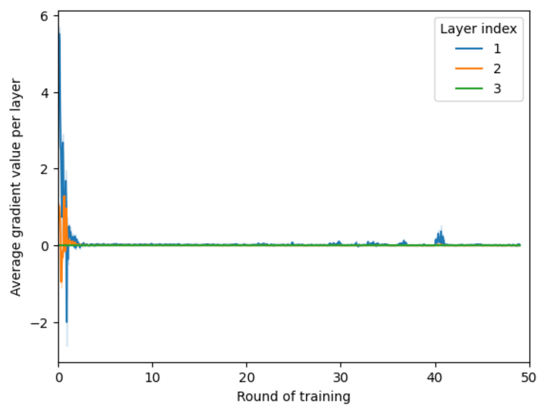

Figure 24. Average gradient value per layer for hierarchical FL in the baseline scenario

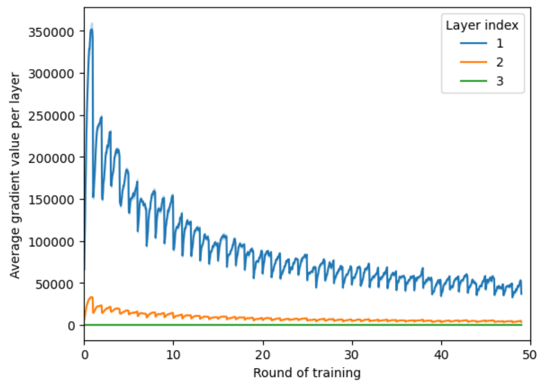

Figure 25. Average gradient value per layer for hierarchical FL in the exploding gradient scenario

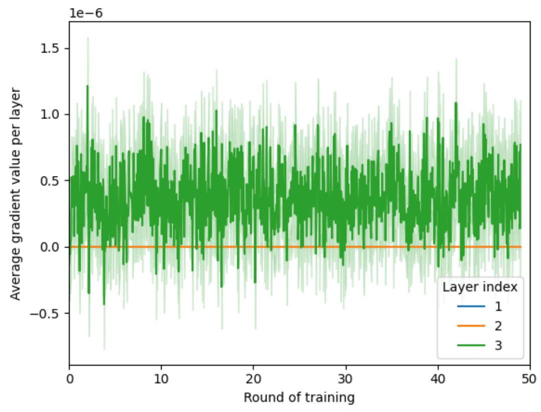

Figure 26. Average gradient value per layer for hierarchical FL in the vanishing gradient scenario

We propose and test a potential monitoring framework designed for the early detection of such issues. We confirm its ability to enable easy differentiation between scenarios of vanishing, exploding, and stable gradient in centralized and hierarchical FL systems with the assumption of IID data. Along with an analysis focused on the visual clarity of our results, we investigate the possibility of a more communicationally and computationally efficient approach by including only the measurements conducted after the last local iteration.

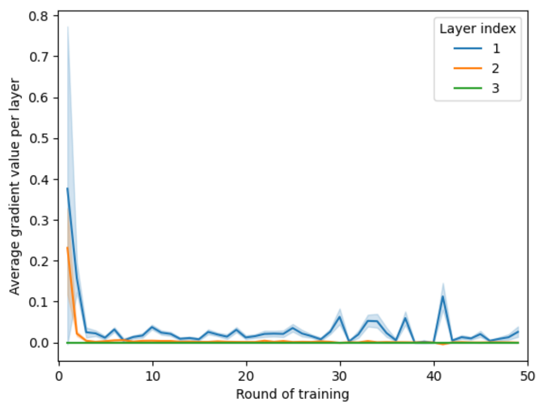

Figure 27. Average gradient value per layer measured for the last local iteration for hierarchical FL in the baseline scenario

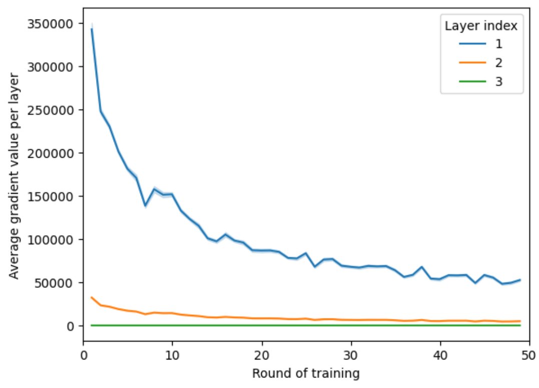

Figure 28. Average gradient value per layer measured for the last local iteration for hierarchical FL in the exploding gradient scenario

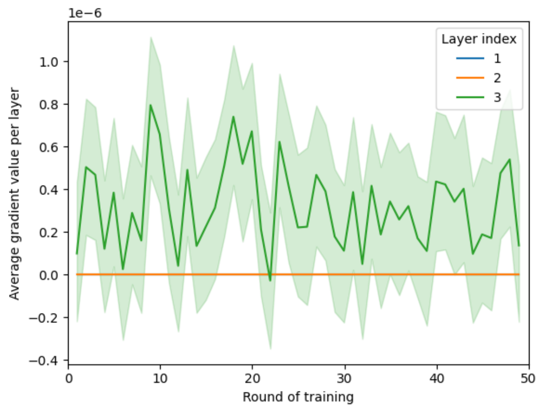

Figure 29. Average gradient value per layer measured for the last local iteration for hierarchical FL in the vanishing gradient scenario

Based on that, we introduce a joint tool including the measurement of GSC and average gradient per layer to enable lightweight and comprehensive gradient monitoring. We include GSC to enable easy visualization of relative gradient stability in the context of previous runs, as well as average gradient per layer to allow a definite diagnosis of the current run as affected by the problem of vanishing or exploding gradients. In the case of hierarchical FL, both should be computed only for the last local iteration each round.

Our work affirms the need to further examine the efficacy of existing tools designed for monitoring in diagnostics in FL systems due to their complex, distributed nature and unique problems such as client dropout or diverging client distributions. An interesting research area we would like to shed light on can also be found in testing tools like the metric suite described in this work in environments simulating varying, heterogenous sets of obstacles, including the aforementioned client dropout, differences in local data distribution, and bad hyper-parameter selection. Future work can also consider the inclusion of other potentially suitable metrics, such as the Nonlinearity Coefficient [30], which is an evolution of the Gradient Scale Coefficient.

Notes

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

AUTHORS

Karolina Bogacka[*] – Warsaw University of Technology, Plac Politechniki 1, 00-661 Warszawa, Poland, e-mail: karolina.bogacka.dokt@pw.edu.pl.

Anastasiya Danilenka – Warsaw University of Technology, Plac Politechniki 1, 00-661 Warszawa, Poland, e-mail: anastasiya.danilenka.dokt@pw.edu.pl.

Katarzyna Wasielewska-Michniewska – Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warszawa, Poland, e-mail: katarzyna.wasielewska@ibspan.waw.pl.

[*]Corresponding author

ACKNOWLEDGEMENTS

The work of Kalina 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.

References

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

[2] A. Bellet, A. Kermarrec, and E. Lavoie, "D-cliques: Compensating noniidness in decentralized federated learning with topology", *CoRR*, vol. abs/2104.07365, 2021.

[3] K. Bogacka, A. Danilenka, and K. Wasielewska-Michniewska, "Diagnosing machine learning problems in federated learning systems: A case study". In: M. Ganzha, L. Maciaszek, M. Paprzycki, and D. Ślęzak, eds., *Proceedings of the 18th Conference on Computer Science and Intelligence Systems*, vol. 35, 2023, 871–876, 10.15439/2023F722.

[4] Q. Cheng and G. Long, "Federated learning operations (flops): Challenges, lifecycle and approaches". In: *2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2022, 12–17, 10.1109/TAAI57707.2022.00012.

[5] L. Chou, Z. Liu, Z. Wang, and A. Shrivastava, "Efficient and less centralized federated learning", *CoRR*, vol. abs/2106.06627, 2021.

[6] A.-I. Consortium. "D7.2 Pilot Scenario Implementation – First Version", 2022.

[7] X. Du, X. Chen, J. Cao, M. Wen, S.-C. Cheung, and H. Jin, "Understanding the bug characteristics and fix strategies of federated learning systems". In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA, 2023, 1358–1370, 10.1145/3611643.3616347.

[8] 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.

[9] H. Eichner, T. Koren, H. B. McMahan, N. Srebro, and K. Talwar, "Semi-cyclic stochastic gradient descent", *CoRR*, vol. abs/1904.10120, 2019.

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

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

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks". In: Y. W. Teh and M. Titterington, eds., *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, Chia Laguna Resort, Sardinia, Italy, 2010, 249–256.

[13] F. Godin, J. Degrave, J. Dambre, and W. De Neve, "Dual rectified linear units (drelus): A replacement for tanh activation functions in quasi-recurrent neural networks", *Pattern Recognition Letters*, vol. 116, 2018, 8–14.

[14] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?". In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 31, 2018.

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

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

[17] 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.

[18] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (mlops): Overview, definition, and architecture", *IEEE Access*, vol. 11, 2023, 31866–31879, 10.1109/ACCESS.2023.3262138.

[19] 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.

[20] A. Li, R. Liu, M. Hu, L. A. Tuan, and H. Yu, "Towards interpretable federated learning", *arXiv preprint arXiv:2302.13473*, 2023.

[21] 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, 2022, 2291–2303, 10.1109/TPDS.2021.3137321.

[22] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection", *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, 2023, 3347–3366, 10.1109/TKDE.2021.3124599.

[23] 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.

[24] 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.

[25] 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.

[26] L. Meng, Y. Wei, R. Pan, S. Zhou, J. Zhang, and W. Chen, "Vadaf: Visualization for abnormal client detection and analysis in federated learning", *ACM Trans. Interact. Intell. Syst.*, vol. 11, no. 3–4, 2021, 10.1145/3426866.

[27] M. A. Mercioni and S. Holban, "The most used activation functions: Classic versus current". In: *2020 International Conference on Development and Application Systems (DAS)*, 2020, 141–145.

[28] 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, 2022, 55–66, 10.1109/TNSE.2021.3053588.

[29] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*, 2013, 1310–1318.

[30] G. Philipp, "The nonlinearity coefficient-a practical guide to neural architecture design", *arXiv preprint arXiv:2105.12210*, 2021.

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

[32] M. Roodschild, J. Gotay Sardiñas, and A. Will, "A new approach for the vanishing gradient problem on sigmoid activation", *Progress in Artificial Intelligence*, vol. 9, no. 4, 2020, 351–360.

[33] Y. Shi, Y. E. Sagduyu, and T. Erpek. "Federated learning for distributed spectrum sensing in nextg communication networks", 2022.

[34] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition", *Neural Networks*, vol. 32, 2012, 323–332, https://doi.org/10.1016/j.neunet.2012.02.016, Selected Papers from IJCNN 2011.

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

[36] 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.

[37] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays. "Applied federated learning: Improving google keyboard query suggestions", 2018.

[38] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates". In: J. Dy and A. Krause, eds., *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, 5650–5659.

[39] 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, 2021, 3790–3804, 10.1109/JSAC.2021.3118423.